Functional Optimization of Fluidic Devices with Differentiable Stokes Flow

TAO DU, MIT CSAIL
KUI WU, MIT CSAIL
ANDREW SPIELBERG, MIT CSAIL
WOJCIECH MATUSIK, MIT CSAIL
BO ZHU, Dartmouth College
EFTYCHIOS SIFAKIS, University of Wisconsin-Madison

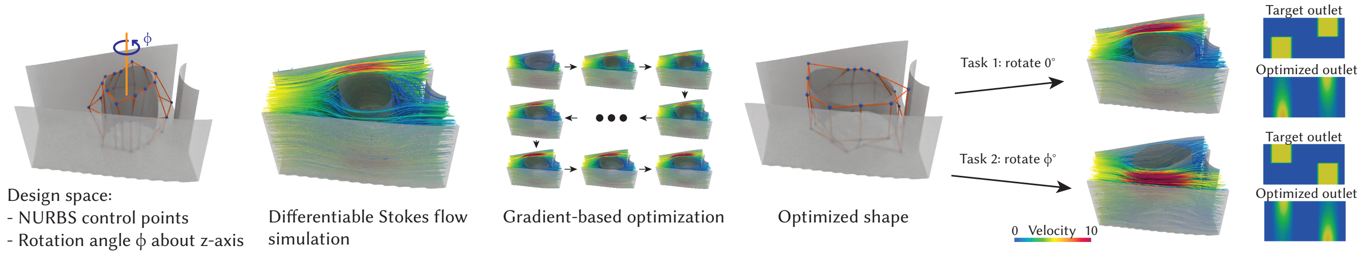

Fig. 1. *Our system automates the design of fluidic devices with differentiable stokes flow. Given a parameterized design in the form of NURBS surfaces or curves (leftmost) that separate rigid boundaries from fluid flow, we employ a Stokes flow (second from left) that evaluates the performance of this design. The flow is differentiable and gradients can be quickly evaluated, enabling gradient-based optimization (center) of the control points, and thus, the boundary. The optimized design (rightmost) can be specified to operate in one configuration or several. This example features an optimized fluidic rotational switch that shifts flow from the top outlet path to the bottom outlet path when turned.*

We present a method for performance-driven optimization of fluidic devices. In our approach, engineers provide a high-level specification of a device using parametric surfaces for the fluid-solid boundaries. They also specify desired flow properties for inlets and outlets of the device. Our computational approach optimizes the boundary of the fluidic device such that its steady-state flow matches desired flow at outlets. In order to deal with computational challenges of this task, we propose an efficient, differentiable Stokes flow solver. Our solver provides explicit access to gradients of performance metrics with respect to the parametric boundary representation. This key feature allows us to couple the solver with efficient gradient-based optimization methods. We demonstrate the efficacy of this approach on designs of five complex 3D fluidic systems. Our approach makes an important step towards practical computational design tools for high-performance fluidic devices.

CCS Concepts: • **Computing methodologies** → **Physical simulation**.

Additional Key Words and Phrases: Physically-based simulation, fluid simulation, computational design optimization

Authors' addresses: Tao Du, MIT CSAIL, taodu@csail.mit.edu; Kui Wu, MIT CSAIL, kuiwu@csail.mit.edu; Andrew Spielberg, MIT CSAIL; Wojciech Matusik, MIT CSAIL, wojciech@csail.mit.edu; Bo Zhu, Dartmouth College, bo.zhu@dartmouth.edu; Eftychios Sifakis, University of Wisconsin-Madison, sifakis@cs.wisc.edu.

ACM Reference Format:
Tao Du, Kui Wu, Andrew Spielberg, Wojciech Matusik, Bo Zhu, and Eftychios Sifakis. 2020. Functional Optimization of Fluidic Devices with Differentiable Stokes Flow. *ACM Trans. Graph.* 39, 6, Article 197 (December 2020), 15 pages. https://doi.org/10.1145/3414685.3417795

1 INTRODUCTION

Fluidic devices are key building blocks for a variety of ubiquitous products, including medical diagnostic devices, filtration systems, bioreactors, internal combustion engines, hydraulic actuators, and even cooling manifolds for GPUs. However, designing complex fluidic devices is challenging as it requires expert knowledge and typically many trial-and-error iterations. These challenges promote the importance of finding computational strategies for simulating and designing these structures. Unfortunately, such approaches are challenging. Brute-force, high-resolution, physics-based simulations of fluidic systems are inherently slow and highly sensitive to geometric configurations and initial conditions, limiting progress in methods for computationally designing fluidic devices with high resolution and complex functions. Furthermore, performance-driven design methods (also often referred to as inverse methods) require using an expensive fluid simulation *within* a numerical optimization method. This effectively makes current approaches for performance-driven optimization impractical.

In this work, we present a first step toward functionally optimizing the design of fluidic devices, focusing on the more tractable *Stokes flow*, which is well-suited for the behaviors of desired fluidic functionality. Stokes flow assumes that fluid velocities are slow and fluid viscosity is relatively large (the Reynolds number Re $\ll 1$). Additionally, in our approach, we use a parametric shape representation of a fluidic system – fluid-solid boundaries are represented using parametric surfaces. This has the advantage that the design process is intuitive for the designer (*e.g.*, a designer specifies an initial shape). At the core of our approach is a differentiable Stokes flow simulator that efficiently solves not only the fluidic dynamics but also the gradients of the dynamics with respect to design parameters. This capability allows us to use this solver as a building block for gradient-based optimization algorithms when performance objectives (*e.g.*, target fluid flows at inlets or outlets) are specified. Overall, our framework unlocks fast fluid flow simulation and gradient computation, making it amenable to continuous optimization.

Our proposed method shares similarities with topology or shape optimization, the two prominent techniques in engineering practice for functional design of fluidic devices [Alexandersen and Andreasen 2020]. The vast majority of prior methods focus on topology optimization for steady-state laminar flows paired with no-slip boundary conditions only [Behrou et al. 2019; Borrvall and Petersson 2003; Gersborg-Hansen et al. 2005; Lin et al. 2015]. While topology optimization yields a geometrically expressive design space, this combination of rasterized and highly frictional boundaries limits both the realism and the *functional* expressiveness of the optimized designs. A less prevalent but more recent line of research is shape optimization of fluidic devices [Villanueva and Maute 2017; Zhou et al. 2018], which is more related to our method. However, to our best knowledge, existing demonstrations from these papers are still coupled with no-slip boundary conditions only, and discussions on extensions to flexible boundary handling in shape optimization are sparse. Our method is in sharp contrast to prior as it simultaneously accommodates smooth parametric shape representations and handles explicit, versatile boundaries. We focus on spline-based parametric boundaries, which naturally yield smooth flows. Further, such parameterizations are low-dimensional (more tractable), more intuitive to reason about, and guarantee physically fabricable devices (*i.e.*, no floating components) when compared with voxel-based parameterizations. Finally, with the careful treatment of sub-cell discretizations in our method, we support various boundary conditions (*e.g.*, no-slip, traction, or no-separation boundary conditions) that allow the emergence of laminar flows in scenarios where such behavior would be anticipated.

To demonstrate the efficacy of our approach, we run performance-driven optimization for the design of complex 3D fluidic systems, including flow averagers, funnels, superposition gates, twisters, and switches. For each example, an engineer starts by specifying an initial fluid-solid boundary with splines, which are all easily parameterized with fewer than 50 degrees of freedom. The engineer then specifies a fluid flow at the inlets of the system and target fluid flow to be optimized at the system's outlets. For all examples, our approach manages to return an optimized design that significantly improves the performance of the device within less than 50 optimization iterations. Furthermore, we demonstrate in our fluidic switch example that our approach supports multifunctional design optimization over continuously varying input velocity configurations.

To summarize, our paper contributes the following:

- a differentiable Stokes flow simulator with a continuous representation of the fluid-solid interface that naturally fits within an optimization framework;
- a sub-cell discretization paradigm in Stokes flow simulation that supports flexible boundary conditions, including no-slip, traction, and no-separation boundaries;
- an optimization pipeline for computational design of multifunctional fluidic devices with continuously varying input velocity configurations.

2 RELATED WORK

Fluid simulation. Simulation of fluid flows has been a staple of physics-based animation, relying predominantly on the Navier-Stokes equations to capture the dynamics of motion in media such as smoke [Fedkiw et al. 2001; Stam 1999] and water [Enright et al. 2002]. Several such methods are based on finite-difference discretizations on regular Cartesian grids, often with a staggered placement of state variables. Level-set methods [Osher and Fedkiw 2003] have been used widely to solve interface problems on a Cartersian grid, in conjunction with the numerical schemes to treat the boundary such as the Ghost Fluid Method [Fedkiw et al. 1999] and variational interpolation [Batty et al. 2007]. Explicit boundary discretizations, such as embedded surface meshes, show their unique merits in modeling the sub-cell geometry and enforcing precise boundary conditions [Azevedo et al. 2016; Schroeder et al. 2012]. These embedded discretizations of the variational type, are focused on handling Dirichlet boundaries [Hellrung et al. 2012; Zhu et al. 2012], which inspired our discretization for solving steady-state flow problems. These discretizations can conveniently accommodate adaptive resolution [Ando et al. 2013] and flows in containers with deforming geometry [Feldman et al. 2005]. Accommodation of changing geometry of the fluid container is also addressed in grid-based techniques that draw inspiration from Arbitrary Lagrangian-Eulerian (ALE) techniques [Ibayashi et al. 2018]. Notably most such works that target evolving fluid domains are focused towards dynamic simulation rather than stationary flows. Steady-state flows, and especially Stokes fluids, have received occasional attention within the graphics literature, in applications related to fluid control [Bhattacharya et al. 2012], simulation of highly viscous media such as paint [Baxter et al. 2004], or as a complement to an unsteady-flow solver for viscous liquids [Larionov et al. 2017].

In terms of the fluid model, besides prior efforts on Stokes flow, our work is also related to methods on simulating nonlinear compressible flows [Kwatra et al. 2009] and viscoplastic materials [Stomakhin et al. 2014] but is different from them: instead of solving the Navier-Stokes equations with explicit pressure terms, we exploit the analogy between Stokes flows and linear elasticity to simulate differentiable, quasi-incompressible Stokes flow without the need for solving the pressure term explicitly. The idea of drawing the analogy between fluids and elastic solids for fluid simulation can also be found in Ferstl et al. [2014], which simulates fluids on an adaptive octree grid using a hexahedral finite element discretization. Although both Ferstl et al. [2014] and our method leverages finite element discretization and elastic solvers for fluid simulation, their approach focuses on forward simulation only. Meanwhile, we develop differentiable flow simulation with comprehensive discussions on gradient derivation.

Fluid control and visualization. Our pipeline for the optimization of fluidic devices has some degree of thematic overlap with fluid control [McNamara et al. 2004; Pan et al. 2013; Raveendran et al. 2012], which has been used broadly in animation applications. In particular, our method shares similarities with Eckert et al. [2019] which optimizes external forces in a fluid system in order to match its behavior to real-world thick smoke. Our focus differs from these prior fluid control papers in that we consider the effect of the geometry of the fluid container as the sole factor influencing the resulting flow. We emphasize steady-state flows and optionally consider multi-objective optimizations under different scenarios of user-imposed boundary conditions. Another thematically related thread of prior research targets interactive visualization of fluids under interactive user manipulation of solid boundaries surrounding the flow [Umetani and Bickel 2018], although our pipeline is more explicitly geared towards active optimization of such designs in steady-state flow scenarios. Data-driven synthesis techniques based on neural networks [Chu and Thürey 2017; Kim et al. 2019] can also produce parametric generative models of fluid flows. Compared to these methods, our first-principles-based approach is not reliant on a comprehensive training corpus and can discover new designs not exemplified in training samples. Finally, we share inspiration from the growing body of recent research on differentiable simulators [Holl et al. 2020; Hu et al. 2019; Li et al. 2019; Liang et al. 2019; Schenck and Fox 2018] which are emerging as a powerful tool for automated design and control applications.

Fluid system optimization. In practice, the most prevalent technique for fluid system optimization is topology optimization [Deaton and Grandhi 2014; Rozvany 2009; Sigmund and Maute 2013]. Beginning with the pioneering work of Borrvall and Petersson [2003], vast literature has been devoted to the optimization of fluid systems, including Stokes flow [Aage et al. 2008; Challis and Guest 2009; Gersborg-Hansen et al. 2005; Guest and Prévost 2006], steady-state flow [Zhou and Li 2008], weakly compressible flow [Evgrafov 2006], fluid-structure interaction (FSI) [Andreasen and Sigmund 2013; Casas and Pavanello 2017; Yoon 2010], aerodynamics [Maute and Allen 2004], and animation [McNamara et al. 2004], to name a few. Some recent work has started to study dynamic and statistical features such as turbulence [Dilgen et al. 2018a,b; Papoutsis-Kiachagias and Giannakoglou 2016]. The development of topology optimization algorithms to explore the dynamic characteristics driven by various fluids remains unexplored due to the complexities regarding both the simulation and optimization.

Unlike these prior efforts, our method chooses parametric shapes as the design space as opposed to voxel grids in topology optimization. While a parametric representation limits the space of possible solutions (*e.g.*, topological structures cannot appear/disappear), this representation also has significant advantages. First, handling of the fluid-solid boundary is accurate and efficient. Second, engineers can also provide input on the types of solutions they have in mind. Finally, the computed representation is editable and directly compatible with current CAD systems since conversion from voxels to parametric surfaces is not necessary.

3 SYSTEM OVERVIEW

Our system is visualized in Fig. 2. As input, a user supplies a parameterized level-set geometry, for example, spline curves or NURBS surfaces (Fig. 2.1). These manifolds separate the solid regions from regions with fluid flow. The user further specifies inlet flow velocities at some point on the fluid portion of the grid (fixed boundary conditions) and an objective to optimize. This objective could be, *e.g.*, target flow velocities at certain designated outlet locations.

During optimization, the following quasi-Newton optimization loop is applied: the current design, as defined by the current parameter instantiation, is simulated on a regular grid with explicit handling of different types of boundary conditions (Fig. 2.2). If the performance of the simulated device does not match the desired objectives of the user (Fig. 2.3), the objective function is differentiated through the simulation with respect to the design parameters to produce a gradient (Fig. 2.4), which is then used to improve the design (back to Fig. 2.1). Otherwise, the optimization is terminated with a successful design (Fig. 2.5).

The remainder of this paper is organized as follows: first, we describe the underlying physical assumptions governing our models, such as constitutive material models and boundary conditions, along with the continuous Partial Differential Equation (PDE) formulation of the physics (Sec. 4). Next, we describe how we discretize these continuous equations into a finite element form that can be simulated (Sec. 5). We then describe the forward simulation (Sec. 6) and gradient computation and optimization framework (Sec. 7), before finally presenting results and discussions (Sec. 8).

4 GOVERNING PARTIAL DIFFERENTIAL EQUATIONS

Since our work targets shape optimization of structures that modulate the flow properties of liquid media, we first present the mathematical model we have adopted for the governing equations of such fluid flows. Given their ubiquity in computational physics and computer graphics, established models of fluid flow such as the incompressible Euler equations for inviscid flows or, more generally, the Navier-Stokes equations for fluids with nontrivial viscosity [Bridson 2015] would be natural choices. However, given the difficulty of the continuous optimization in the inverse design problem at hand, we consciously restrict our material model to a narrower set of smoother, more well-behaved fluid behaviors. First, we specifically seek to model *steady-state flows* in order to avoid transient effects and avoid time dependencies in our optimization task. Second, in order to avoid local minima associated with non-unique solutions as well as boost the speed and conditioning of the optimization scheme, we employ the linearized form of the steady-state Navier-Stokes equations, also known as *Stokes flow* [Lautrup 2004].

Incompressible Stokes equations. We initially review the PDE form of the Stokes system and describe the boundary value problem that would be typically formulated for flow scenarios as in our target application. Let $\Omega \subset \mathcal{R}^d$ ($d = 2$ or 3) be a domain bounded by a smooth boundary Γ. In the standard Eulerian perspective, the Stokes

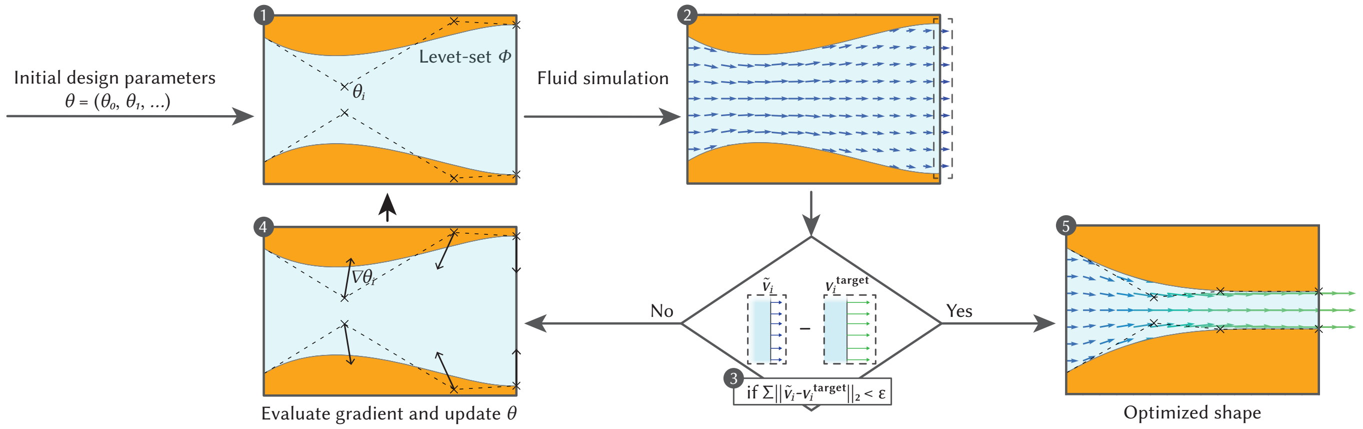
Initial design parameters
θ = (θ₀, θ₁, ...)
Levet-set Φ
θᵢ
Fluid simulation
∇θᵣ
No
Yes
ṽᵢ
vᵢ^target
if Σ||ṽᵢ-vᵢ^target||₂ < ε
Evaluate gradient and update θ
Optimized shape

Fig. 2. *An overview of our system: 1: the design parameter $\boldsymbol{\theta}$ (either explicitly given or randomly initialized) determines the fluid-solid boundaries in the design problem. 2: for any given $\boldsymbol{\theta}$, we simulate the Stokes flow in the fluidic domain and enforce different types of boundary conditions explicitly. 3: we then evaluate the loss function on the resulting velocity field and test it against the termination condition. 4: if the result is not optimal, we differentiate the loss with respect to $\boldsymbol{\theta}$ and its gradient is applied in a gradient-based local optimization method to update the design parameter. 5: the algorithm terminates with an optimized $\boldsymbol{\theta}$ and the corresponding design.*

equations yield a velocity field $\boldsymbol{v} : \Omega \to \mathcal{R}^d$ and a pressure field $p : \Omega \to \mathcal{R}$ as solutions to the PDE system

$$-\eta\Delta\boldsymbol{v}(\boldsymbol{x}) + \nabla p(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}) \qquad \boldsymbol{x} \in \Omega \tag{1}$$
$$\nabla \cdot \boldsymbol{v}(\boldsymbol{x}) = 0 \qquad \boldsymbol{x} \in \Omega \tag{2}$$

where η is the dynamic viscosity and $\boldsymbol{f}(\boldsymbol{x})$ an externally applied force field (e.g. gravity) if applicable. We note that Eqn. (1) is derived from the momentum equation $\nabla \cdot \boldsymbol{T}(\boldsymbol{x}) + \boldsymbol{f}(\boldsymbol{x}) = 0$ after substituting the linear constitutive law for the stress tensor $\boldsymbol{T}$

$$\boldsymbol{D} = \frac{1}{2}[\nabla\boldsymbol{v} + (\nabla\boldsymbol{v})^T] \tag{3}$$
$$\boldsymbol{T} = 2\eta\boldsymbol{D} - p\boldsymbol{I} = \eta[\nabla\boldsymbol{v} + (\nabla\boldsymbol{v})^T] - p\boldsymbol{I} \tag{4}$$

and using the incompressibility Eqn. (2) to simplify the result; here, $\nabla\boldsymbol{v}$ is the spatial gradient of $\boldsymbol{v}$, $\boldsymbol{D}$ the strain rate tensor, and $\boldsymbol{I}$ the $d \times d$ identity matrix.

Boundary conditions along the boundary Γ may be chosen from several types, according to the intended scenario and application. The most straightforward would be *Dirichlet boundary conditions*

$$\boldsymbol{v}(\boldsymbol{x}) = \boldsymbol{\alpha}(\boldsymbol{x}) \qquad \boldsymbol{x} \in \Gamma_D \tag{5}$$

on any part of the boundary, denoted as Γ_D where we want to have a prescribed velocity profile $\boldsymbol{\alpha}(\boldsymbol{x})$, as in the inlet to the apparatus depicted in Fig. 3 (a). In those cases where we seek to model a highly viscous contact layer, a *no-slip* zero-Dirichlet boundary condition $\boldsymbol{v}(\boldsymbol{x}) = \boldsymbol{0}$ would also be enforced along the surface of the container wall; this is used in only a minority of our examples, but is certainly an option within our framework.

The remaining types of boundary conditions encountered in our framework involve the *traction* vector $\boldsymbol{\tau}(\boldsymbol{x}) = \boldsymbol{T}(\boldsymbol{x}) \cdot \boldsymbol{n}(\boldsymbol{x})$, defined on a boundary location $\boldsymbol{x} \in \Gamma$ with outward-pointing normal vector $n(\boldsymbol{x})$. A *traction condition*

$$\boldsymbol{\tau}(\boldsymbol{x}) = \boldsymbol{\beta}(\boldsymbol{x}) \qquad \boldsymbol{x} \in \Gamma_T \tag{6}$$

would typically be used in outlets of our flow device where, instead of prescribing a flow profile, we would provide an externally applied force along the associated boundary (*i.e.* a cross-section of the fluid container) that intends to either impede or boost the flow. An example would be a permeable membrane affixed to an outlet that seeks to impede the flow by applying a resistive force. The specific type of boundary condition used in all our examples is $\boldsymbol{\tau}(\boldsymbol{x}) = \boldsymbol{0}$ (equivalently, $\boldsymbol{\beta}(\boldsymbol{x}) = \boldsymbol{0}$) which we would refer to as an *open boundary* condition and corresponds to the flow being allowed to transit through the domain boundary freely, without either being impeded or boosted by any external influence (see Fig. 3 (a)).

The final type of boundary condition we optionally employ in our framework is a *no-separation* (and, in essence also *no-friction*) boundary condition along the walls of the enclosing container. This mixed boundary condition is captured in the following equations

$$\boldsymbol{v}(\boldsymbol{x}) \cdot \boldsymbol{n}(\boldsymbol{x}) = 0 \qquad \boldsymbol{x} \in \Gamma_S \tag{7}$$
$$\boldsymbol{\tau}_t(\boldsymbol{x}) = \boldsymbol{0} \qquad \boldsymbol{x} \in \Gamma_S \tag{8}$$

where the first component is conveyed by the scalar (1D) condition in Eqn. (7) and dictates that the flow should be parallel to the container wall (with $\boldsymbol{n}(\boldsymbol{x})$ being the normal vector at a boundary point $\boldsymbol{x} \in \Gamma_S$); this suggests that the flow will neither separate from the container, nor will it penetrate into it. Eqn. (8) dictates that the *tangential* component $\boldsymbol{\tau}_t$ of the traction vector should be equal to zero; this constraint has $(d-1)$ dimensions as it is projected on the tangential plane at each boundary location. Intuitively, this condition suggests that the fluid flow is not subject to any frictional forces that would impede its tangential motion; when combined with the no-separation condition this yields the same number of d equations per boundary point as in other types of boundary conditions. We employ this type of boundary condition broadly (albeit, not exclusively) in our examples, as it enables the emergence of the type of laminar steady-state flows that we would intuitively expect with a friction-free contact layer.

Relation to linear elasticity. Well-known parallels exist between the Stokes problem and the PDEs of *linear elasticity*, which are broadly used in shape and topology optimization applications. We should emphasize that these analogies – stemming from the fact that both equations emerge from directly congruous conservation laws – are despite the fact that the underlying state variable has a different physical meaning for fluids versus elastic solids. In fluids, the PDE is defined over a *velocity* field, and in elastic media, over a *displacement* field.

Although we will demonstrate this analogy in its most stark form in the limit of *incompressible* linear elastic materials, we will start our review of this relation from the standard (i.e. compressible) linear elasticity PDE. For an elastic medium whose shape change is encoded via a deformation map $\boldsymbol{x}(\boldsymbol{X}) : \Omega \longrightarrow \mathcal{R}^d$ (where $\boldsymbol{X}$ are material/undeformed coordinates and $\boldsymbol{x}$ the corresponding spatial/deformed locations), we define the *displacement* field as $\boldsymbol{u}(\boldsymbol{X}) = \boldsymbol{x}(\boldsymbol{X}) - \boldsymbol{X}$, and subsequently define the small-strain tensor $\boldsymbol{\epsilon}$ and Cauchy stress $\boldsymbol{\sigma}$ from a linear stress-strain relationship

$$\boldsymbol{\epsilon} = \frac{1}{2}[\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^T] \tag{9}$$

$$\boldsymbol{\sigma} = 2\mu\boldsymbol{\epsilon} + \lambda\,\mathrm{tr}(\boldsymbol{\epsilon})\boldsymbol{I} \tag{10}$$

where μ, λ here are the Lamé coefficients of the elastic material. Substituting the stress tensor $\boldsymbol{\sigma}(\boldsymbol{x})$ into the momentum equation $\nabla \cdot \boldsymbol{\sigma}(\boldsymbol{x}) + \boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$ (where $\boldsymbol{f}(\boldsymbol{x})$ are the external forces, if any) now yields the PDE of linear elasticity [Sifakis and Barbic 2012]:

$$-\mu\Delta\boldsymbol{u}(\boldsymbol{x}) - (\mu+\lambda)\nabla[\nabla\cdot\boldsymbol{u}(\boldsymbol{x})] = \boldsymbol{f}(\boldsymbol{x}) \qquad \boldsymbol{x} \in \Omega. \tag{11}$$

The relation to the Stokes equations will start becoming more apparent if we consider an almost incompressible material for which the value of λ is significantly larger than that of μ; although the solution of the PDE evolves smoothly and continuously as the parameter λ asymptotically reaches infinity, the exact form of the PDE in Eqn. (11) would not be the ideal way to express it, due to the unbounded coefficients involved. Instead, we can derive a better behaved, equivalent system in the spirit of mixed formulations [Brezzi and Fortin 2012; Zhu et al. 2010], by introducing a new, auxiliary state variable $r(\boldsymbol{x})$ defined as

$$r(\boldsymbol{x}) = -(\mu+\lambda)\nabla\cdot\boldsymbol{u}(\boldsymbol{x}). \tag{12}$$

By substituting this expression into Eqn. (11) and rearranging terms in Eqn. (12), we arrive at the following equivalent PDE system for compressible linear elasticity

$$-\mu\Delta\boldsymbol{u}(\boldsymbol{x}) + \nabla r(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}) \qquad \boldsymbol{x} \in \Omega \tag{13}$$

$$\nabla\cdot\boldsymbol{u}(\boldsymbol{x}) + \frac{1}{\mu+\lambda}r(\boldsymbol{x}) = 0 \qquad \boldsymbol{x} \in \Omega. \tag{14}$$

Once we have arrived at this form, the analogy between the Stokes equations and the above equations of linear elasticity start becoming more apparent. We highlight the following observations:

- It should be clarified that any similarities between the two governing laws are restricted to the form of their PDEs, while the underlying state variables are semantically distinct. Specifically, η, $\boldsymbol{v}$, and p in Eqns. (1, 2) play the same role as μ, $\boldsymbol{u}$, and r in Eqns. (13, 14), although their physical meanings are quite different, *e.g.*, in Stokes flow, $\boldsymbol{v}$ is a velocity field, where in elasticity $\boldsymbol{u}$ refers to a field of elastic displacements. These semantic differences do not prevent us, however, from exploiting the similarities at the PDE level.
- It is known [Brezzi and Fortin 2012; Olshanskii et al. 2009] that the reformulated system in Eqn. (13) and (14) is smooth (and also, elliptic) and remains well behaved in the asymptotic limit $\lambda \longrightarrow \infty$ when the coefficient of $r(\boldsymbol{x})$ in Eqn. (14) will merely vanish. The solution to the PDE system, itself, will smoothly and uniformly converge to a limit behavior as we asymptotically approach strict incompressibility.
- If we specifically consider the asymptotic case of strict incompressibility ($\lambda \longrightarrow \infty$), then Eqn. (13) and (14) reduce *exactly* to the Stokes equations as stated in the prior paragraph.

The analogy (and, actually, equivalence) of the linear elasticity and Stokes PDEs would not be complete if we did not also address the form that the respective *boundary conditions* that the two sets of equations might employ. Dirichlet conditions, of course, are equally applicable to both formulations. Those boundary conditions, however, that involve the stress tensor $\boldsymbol{T}$ in Stokes flow and $\boldsymbol{\sigma}$ in linear elasticity require special attention. Taking the trace of Eqn. (9) yields $\mathrm{tr}(\boldsymbol{\epsilon}) = \nabla\cdot\boldsymbol{u}$; using this equality and the definition of r in Eqn. (12) allows us to rewrite the stress tensor from Eqn. (10) as

$$\begin{aligned}\boldsymbol{\sigma} &= \mu[\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^T] - \frac{\lambda}{\mu+\lambda}r\boldsymbol{I} \\ &= \mu[\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^T] - 2\nu r\boldsymbol{I}\end{aligned} \tag{15}$$

where $\nu = \frac{\lambda}{2(\mu+\lambda)}$ is the Poisson's ratio that approaches the value 0.5 in the incompressible limit. Once again, we observe that in the incompressible limit, the stress tensors in both linear elasticity and Stokes converge to the same limit form; as a consequence, so would any traction boundary conditions that would derive from this stress tensor. This demonstrates the asymptotic equivalence of Stokes and linear elasticity at the near-incompressible limit.

Our model: quasi-incompressible Stokes. The aforementioned relation of Stokes and linear elasticity has previously been leveraged primarily to develop discretization and solution schemes for incompressible or near-incompressible elasticity that draw inspiration from established methods for Stokes [Gaspar et al. 2008; Zhu et al. 2012]. However, directly pursuing a discretization of the Stokes problem has its own subtleties; due to the incompressibility constraint, the associated discretizations – and especially variational formulations – take the form of saddle point problems, restricting somewhat the options for associated numerical solvers. Boundary treatment at sub-element precision is relatively nontrivial, especially if certain numerical properties of the discretization (e.g. symmetry) are to be preserved [Zhu et al. 2010].

We have thus decided, in this initial venture into shape optimization involving fluid flows, to move in the opposite direction, and use the equations of linear elasticity in the *near-incompressible* (e.g. $\nu \approx 0.49$) but not strictly incompressible regime. We choose to use the common form of the linear elastic model in Eqn. (11) as opposed to the pressure-augmented system (Eqn. (13) and (14)), and also use the corresponding expression for the stress tensor, as in Eqn. (10) in the formulation of traction or no-separation/no-friction

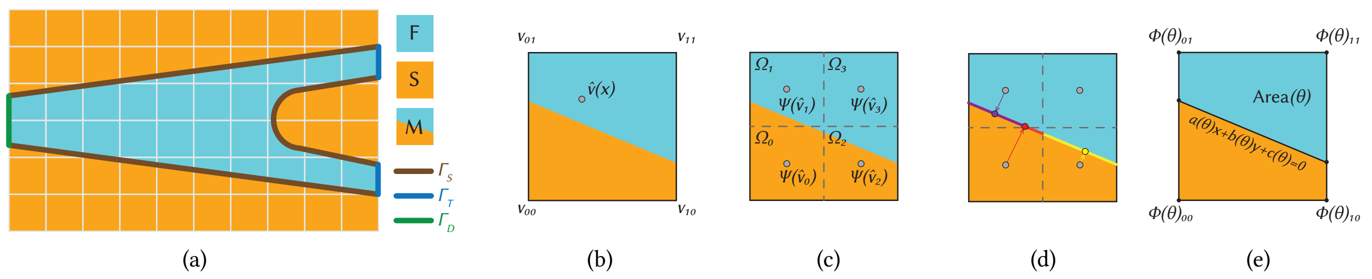

Fig. 3. *(a) the entire 2D space is discretized into fluid, solid, and mixed cells, with three types of boundaries as discussed in Sec. 4. (b) fluid velocities $\boldsymbol{v}_i$ are stored on grid nodes. (c) the fluid energy density is evaluated on different quadrature points and integrated over the entire cell by multiplying by the fluid occupied area at each subcell. (d) the Dirichlet boundary condition is enforced by integrating the values over the linearized interface, with the quadrature points obtained from the projection of quadrature points in (c) onto the interface. (e) all geometric information is defined by design parameters $\boldsymbol{\theta}$ and linearized within each subcell.*

boundary conditions. The new governing equation for our quasi-incompressible Stokes flow model can be obtained by replacing μ and $\boldsymbol{u}$ in Eqn. (11) with the dynamic viscosity η and the velocity field $\boldsymbol{v}$ from Stokes flow:

$$-\eta\Delta\boldsymbol{v}(\boldsymbol{x}) - \frac{\eta}{1-2\nu}\nabla[\nabla\cdot\boldsymbol{v}(\boldsymbol{x})] = \boldsymbol{f}(\boldsymbol{x}) \qquad \boldsymbol{x} \in \Omega. \tag{16}$$

Similarly, the traction tensor $\boldsymbol{\tau} = \boldsymbol{T}\cdot\boldsymbol{n}$ used by the boundary conditions is implemented with the following stress tensor:

$$\boldsymbol{T} = \eta[\nabla\boldsymbol{v} + (\nabla\boldsymbol{v})^T] + \eta(\frac{2\nu}{1-2\nu}\nabla\cdot\boldsymbol{v})\boldsymbol{I}. \tag{17}$$

In both equations, the Poisson's ratio ν controls the incompressibility of our Stokes model. When $\nu \longrightarrow 0.5$, these two equations converge back to Eqns. (1, 2, 4). Note that λ in the linear elasticity equations has been replaced with $\frac{2\eta\nu}{1-2\nu}$ from the relation $\nu = \frac{\lambda}{2(\eta+\lambda)}$.

While we choose to model quasi-incompressible Stokes with an analogy between linear elasticity and Stokes, it is worth pointing out that using the saddle point formulation for discretizing the truly incompressible Stokes flow is still a viable technique. In fact, it would be recommended when paired with an iterative sparse linear solver like Preconditioned Conjugate Gradient (PCG) or multigrid methods. We stress that we opt to use the quasi-incompressible formulation due to our reliance on direct sparse solvers, whose advantages over iterative solvers will become evident in gradient computation (Sec. 7). Furthermore, there is a much higher degree of comfort and experience in standard topology optimization with "stock" linear elasticity, while Stokes systems are not as widespread.

5 NUMERICAL DISCRETIZATION

We discretize our governing equations on a Cartesian background grid that embeds the geometry of the fluid cavity as in Fig. 3 (a) (as opposed to using a mesh that conforms to the boundary of the fluid container). We employ a collocated discretization where all components of the velocity field are stored at the same locations, at the nodes of the Cartesian grid (as opposed to staggered, marker-and-cell (MAC) grid discretizations), and since we use Eqn. (16) for our quasi-incompressible Stokes fluid, there is no need to involve any "pressure" state variables in our formulation. Using a variational approach, we can express boundary conditions at a sub-grid resolution while only storing variables at regular grid-node locations. Again, we stress that due to the strong resemblance between Eqn. (16) and its linear elasticity counterpart Eqn. (11), the numerical discretization paradigm to be discussed in this section is essentially the commonly used discretization scheme in linear elasticity in disguise, allowing practitioners to reuse existing implementations in standard topology optimization with very little extra effort.

Variational form and embedded traction boundaries. We initially focus on the quasi-incompressible Stokes problem in a domain $\Omega \in \mathcal{R}^d$, under *traction* boundary conditions, stated as follows

$$-\nabla\cdot\boldsymbol{T} = -\eta\Delta\boldsymbol{v}(\boldsymbol{x}) - \frac{\eta}{1-2\nu}\nabla[\nabla\cdot\boldsymbol{v}(\boldsymbol{x})] = \boldsymbol{f}(\boldsymbol{x}) \qquad \boldsymbol{x} \in \Omega \tag{18}$$

$$\boldsymbol{T}\cdot\boldsymbol{n} = \boldsymbol{\beta}(\boldsymbol{x}) \qquad \boldsymbol{x} \in \partial\Omega \tag{19}$$

where the stress tensor $\boldsymbol{T}$ is defined as in Eqn. (17). It is known [Daux et al. 2000; Hughes 2012] that the associated variational formulation of this problem computes the solution via minimization of an energy functional $E[\boldsymbol{v}]$ over all functions $\boldsymbol{v}$ in an appropriate solution space. For our purposes, we define the solution space to be all functions defined by bilinear (2D) or trilinear (3D) interpolation over the cells of the background Cartesian grid, and the associated energy to be minimized is [Daux et al. 2000; Zhu et al. 2012]

$$E[\boldsymbol{v}] = \int_\Omega \Psi[\boldsymbol{v}(\boldsymbol{x})]d\boldsymbol{x} - \int_\Omega (\boldsymbol{v}\cdot\boldsymbol{f})d\boldsymbol{x} - \int_{\partial\Omega}(\boldsymbol{v}\cdot\boldsymbol{\beta})dS \tag{20}$$

where the energy density $\Psi[\boldsymbol{v}]$ is defined as

$$\Psi[\boldsymbol{v}] = \eta\|\boldsymbol{D}[\boldsymbol{v}]\|_F^2 + \frac{\eta\nu}{1-2\nu}[\mathrm{tr}(\boldsymbol{D}[\boldsymbol{v}])]^2$$

with the strain rate $\boldsymbol{D}$ computed from $\boldsymbol{v}$ as in Eqn. (3). We note that in most of our examples we do not use any external forces (e.g. gravity), hence $\boldsymbol{f} = 0$, and only use zero-valued (or open-boundary) traction boundary conditions, thus $\boldsymbol{\beta} = 0$. As a consequence, the last two integrals in Eqn. (20) are zero for our examples. Should it be necessary, however, to incorporate non-zero forces or traction conditions in a different application, these terms can trivially be included in our discretization, and we later discuss how both volume and boundary integrals can be computed via quadrature within our solution space.

Sub-cell energy quadrature. Using bilinear/trilinear interpolation as shown in Fig. 3 (b), our solution space contains all functions of the form

$$\hat{v}(x;\mathcal{V}) = \sum_i v_i \mathcal{N}_i(x) \tag{21}$$

where $\mathcal{N}_i(x)$ is the shape function associated with grid node i, and $\mathcal{V} = \{v_i\}$ collectively represents all nodal velocities in our grid. Using this interpolation, we can also express all derivative quantities as functions of the nodal velocities, by proper manipulation of the shape functions. For example, the entries of the strain rate tensor $\hat{D}(x;\mathcal{V}) = D[\hat{v}(x;\mathcal{V})]$ are evaluated as

$$\hat{D}_{pq} = \frac{1}{2}\sum_i \left[v_i^{(p)} \mathcal{N}_{i,q}(x) + v_i^{(q)} \mathcal{N}_{i,p}(x) \right] \tag{22}$$

where superscripts in parentheses for v indicate coordinate components, and subscripts in shape functions after commas indicate partial derivatives. We can continue this substitution to express the energy density and ultimately the integrated energy in Eqns. (20) as a function of the nodal velocity values. Since $\hat{D}$ is a linear function of nodal velocities, and the energy density Ψ has a quadratic dependence on D, the overall energy $E[\mathcal{V}] = E[\hat{v}]$ will ultimately reduce to a quadratic convex function over the nodal velocities, with the coefficients of this polynomial involving integrals of products of derivatives of the shape functions. All of this is, of course, merely a restatement of the standard finite element discretization approach in a Cartesian lattice [Hughes 2012; Patterson et al. 2012].

The integral in Eqn. (20) can be computed on a per-cell basis; using our assumption that $f = \beta = 0$, we can write

$$E[\mathcal{V}] = \int_\Omega \Psi[\hat{v}(x;\mathcal{V})]dx = \sum_k \underbrace{\int_{\Omega\cap C_k} \Psi[\hat{v}(x;\mathcal{V})]dx}_{E_k[\mathcal{V}]} \tag{23}$$

where the summation is taken over all cells $\{C_k\}$ in our background grid. The per-cell energies E_k fall in one of two categories. For *fully interior cells* (for which $C_k \subset \Omega$) the integral can be computed exactly either via analytic integration (the integrands are low-degree polynomials), or with a 4-point (8-point in 3D) Gauss quadrature; this yields the same stencil that is used in several similar methods [Aage et al. 2017; Bendsoe and Sigmund 2013; Liu et al. 2018]. For *boundary cells* (those that have $C_k \cap \partial\Omega \neq \emptyset$) we must specify a quadrature rule for the partial-cell domain of integration $C_k \cap \Omega$.

We propose a quadrature rule for such boundary cells motivated by the following design objectives: (a) We seek a rule that is as simple as possible, so as to be easily adaptable to scenarios where the boundary location is evolving, as in the context of shape optimization, (b) The rule must give rise to continuous solutions as the boundary evolves, to ensure differentiability of such solution with respect to design parameters, and (c) The rule should have at least a rudimentary degree of accuracy (*e.g.* exactly integrate constant integrands) and be free of common defects. In light of these design traits, we propose a quadrature formula that uses four weighted quadrature points (as shown in Fig. 3 (c) in 2D), placed at the centers of four equal quadrants $C_k^{(0)}, \ldots, C_k^{(3)}$ produced by bisecting the boundary cell along each axis (see Fig. 3 (c)). If we denote by $x_0, \ldots, x_3$ the centers of these quadrants and by $\Omega_0 := C_k^{(0)} \cap \Omega, \ldots, \Omega_3 := C_k^{(3)} \cap \Omega$ the fractions of these quadrants that are interior to the fluid domain Ω, the quadrature rule becomes

$$\int_{\Omega\cap C_k} \Psi[\hat{v}(x;\mathcal{V})]dx \approx \sum_{j=0}^{3} \mathrm{Area}(\Omega_j)\Psi[\hat{v}(x_j;\mathcal{V})]. \tag{24}$$

A similar quadrature rule would naturally be defined in 3D, using eight quadrature points at the center of the octants that a cell is split by bisecting each axis, weighted by the corresponding volume fraction of each that falls inside Ω. It is clear that the quadrature rule integrates constant functions exactly (due to the area factors), and that it would provide for continuously varying solutions as the boundary evolves (as the minimizers of a convex quadratic with continuously varying coefficients). The use of multiple quadrature points (as opposed to a single one, at the centroid of the cell C_k) is mandated in order to avoid hourglassing instabilities in the discretization [McAdams et al. 2011]. However, keeping this quadrature rule simple by only having the area factors dependent on the geometry of Ω greatly simplifies the task of differentiating our flow solution with respect to the design parameters, as we see in Sec. 7. We have found this formulation to be effective and sufficient in our examples, and as we see next it can be used in conjunction with the other boundary condition types we need in our application.

Dirichlet boundary conditions. The formulation of the preceding paragraph is sufficient to accommodate pure *traction* boundary conditions, as the open boundary conditions at the outlet of the fluidic device in Fig. 3 (a) in blue. In addition, we can easily accommodate Dirichlet boundary conditions imposed precisely at grid nodes, by simply setting them to a constant value while minimizing $E[v]$. A more challenging, but essential scenario to accommodate would be the enforcement of Dirichlet conditions on an *embedded boundary* rather than one that is aligned with the grid faces. Such an example would correspond to the lateral edges of the device in Fig. 3 (a) in brown, should a no-slip Dirichlet condition ($v = 0$) have been imposed.

Enforcement of such embedded Dirichlet conditions is not quite straightforward with variational formulations, as opposed to traction conditions (analogous to *Neumann* conditions in the Poisson's equation) which are naturally incorporated into the energy $E[v]$. Possible options such as a "soft" constraint enforcement [Lee et al. 2009; Sifakis et al. 2007] have to contend with ad-hoc constraint stiffnesses, while imposing Dirichlet conditions at zero crossings between the interface and grid edges is known to be questionable in its convergence quality or even the existence of a compliant solution [Bedrossian et al. 2010; Moës et al. 1999]. Instead, we employ a formulation for the enforcement of embedded Dirichlet conditions that leverages a weak formulation of the constraint using an appropriate approximation space [Hellrung et al. 2012; Zhu et al. 2012], that has been successfully used with elliptic PDEs in similar contexts.

Let $v(x) = \alpha(x)$ be the Dirichlet condition we want enforced in a section of the boundary $\Gamma_D \subset \partial\Omega$ that is intersecting grid cells, rather than being aligned with edge boundaries. Following previous work [Bedrossian et al. 2010; Zhu et al. 2012], for each such cell, we enforce the Dirichlet condition in an *averaged* fashion, via the integral constraint

$$\int_{C_k \cap \Gamma_D} \hat{\boldsymbol{v}}(\boldsymbol{x}; \mathcal{V}) d\boldsymbol{x} = \int_{C_k \cap \Gamma_D} \boldsymbol{\alpha}(\boldsymbol{x}) d\boldsymbol{x}$$

which, given the expansion of $\hat{\boldsymbol{v}}$ using the shape functions, becomes

$$\sum_i \boldsymbol{v}_i \int_{C_k \cap \Gamma_D} \mathcal{N}_i(\boldsymbol{x}) d\boldsymbol{x} = \int_{C_k \cap \Gamma_D} \boldsymbol{\alpha}(\boldsymbol{x}) d\boldsymbol{x}. \tag{25}$$

Eqn. (25) is effectively a (d-dimensional) linear equality constraint associated with each boundary cell. The integral of the shape function on the left-hand side is computed analytically *via* a hyperplane approximation to the cell boundary $C_k \cap \Gamma_D$. We construct a best-fit line in 2D (plane in 3D) to this boundary section based on the signed distances from grid nodes to the boundary. We use a quadrature rule to approximate the integral of both the shape function and $\boldsymbol{\alpha}$ over $C_k \cap \Gamma_D$ unless the integral is trivial to compute analytically, e.g., $\boldsymbol{\alpha}(\boldsymbol{x})$ is constant. In our application, only a no-slip, *zero-Dirichlet* boundary condition $\boldsymbol{\alpha}(\boldsymbol{x}) = \boldsymbol{0}$ is employed, hence the integral on the right-hand side is trivially zero. We discuss this quadrature rule in greater detail in Sec. 6 and our supplemental material.

Aggregating all such constraints from all cells that intersect the Dirichlet boundary yields a linear system of constraints $\boldsymbol{Cv} = \boldsymbol{d}$ (for simplicity of notation we use here the symbol $\boldsymbol{v}$ for *all* nodal velocities, in replacement of $\mathcal{V}$). As mentioned before, for no-slip boundaries we would have $\boldsymbol{d} = \boldsymbol{0}$.

No-separation, zero-friction boundaries. Although no-slip conditions can be accommodated as in the previous paragraph, enforcing no-separation boundary conditions combined with a zero tangential component of the traction vector is the norm in our examples, as encoded in Eqns. (7, 8). Similar to the previous paragraph, the projected Dirichlet condition $\boldsymbol{v} \cdot \boldsymbol{n} = 0$ is enforced via an integral constraint

$$\int_{C_k \cap \Gamma_D} \hat{\boldsymbol{v}}(\boldsymbol{x}; \mathcal{V}) \cdot \boldsymbol{n} d\boldsymbol{x} = \sum_i (\boldsymbol{v}_i \cdot \boldsymbol{n}) \int_{C_k \cap \Gamma_D} \mathcal{N}_i(\boldsymbol{x}) d\boldsymbol{x} = 0 \tag{26}$$

which is now just a single scalar constraint (per cell) as shown in Fig. 3 (d), while the zero tangential component of the traction is implicitly enforced from the energy formulation in Eqn. (20). Again, a single constraint system of the form $\boldsymbol{Cv} = \boldsymbol{d}$ can aggregate all boundary conditions other than traction boundaries (which are incorporated in the energy), including (a) node-aligned Dirichlet constraints, (b) embedded Dirichlet constraints, and (c) no-separation conditions.

6 FORWARD SIMULATION

Given the discretization described in Sec. 5, we now provide a means of computing the fluid velocity field $\boldsymbol{v}$ given the boundary shape parameterized by a vector of parameters $\boldsymbol{\theta}$. The specific definition of $\boldsymbol{\theta}$ depends on the type of parametric surfaces. Additionally, if multiple parametric surfaces exist in the problem domain, their parameters are aggregated into a single vector $\boldsymbol{\theta}$. As described in Sec. 5, $\boldsymbol{v}$ is the minimizer of the variational form of the energy in Eqn. (20) with boundary conditions applied. Due to the analogy between Stokes flow and linear elasticity, the post-discretization variational form of the energy, known to be quadratic in $\boldsymbol{v}$, can be defined as $\frac{1}{2}\boldsymbol{v}^T\boldsymbol{K}\boldsymbol{v}$ where $\boldsymbol{K}$ has an equivalent role of the stiffness matrix in linear elasticity. Further, the discretized boundary conditions, already introduced as $\boldsymbol{Cv} = \boldsymbol{d}$, are linear in $\boldsymbol{v}$. Combined, these form a convex quadratic programming (QP) problem, the solution to which describes the fluid velocity

$$\tilde{\boldsymbol{v}}(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{v}} \quad \frac{1}{2}\boldsymbol{v}^T\boldsymbol{K}(\boldsymbol{\theta})\boldsymbol{v} \tag{27}$$

$$\text{s.t.} \quad \boldsymbol{C}(\boldsymbol{\theta})\boldsymbol{v} = \boldsymbol{d}(\boldsymbol{\theta}). \tag{28}$$

Here, the convexity comes from the positive semi-definiteness of the stiffness matrix $\boldsymbol{K}$. The notation $\tilde{\boldsymbol{v}}$ is used to emphasize that this is the *minimizer* of the QP problem, dependent on the parameter $\boldsymbol{\theta}$. Meanwhile, $\boldsymbol{K}, \boldsymbol{C}$, and $\boldsymbol{d}$ are written as functions of $\boldsymbol{\theta}$ as their values are dependent on the location of the solid-fluid interface boundaries. This should be especially obvious in the case of $\boldsymbol{K}$, as in Eqn. (24) we can see that boundary cells contribute to this term by an amount proportional to the area of their region of overlap with Ω. Again, we point out that the external force $\boldsymbol{f}$ and traction condition $\boldsymbol{\beta}$ are ignored for simplicity. Should it be necessary, both of them could be easily added back to Eqn. (27) as a linear term on $\boldsymbol{v}$ with its linear weights dependent on $\boldsymbol{\theta}$.

The remainder of this section is dedicated to describing how $\boldsymbol{K}, \boldsymbol{C}$, and $\boldsymbol{d}$ are computed from $\boldsymbol{\theta}$, concretely describing how to calculate the steady-state flow and laying the groundwork for the gradient computation. Given a set of design parameters $\boldsymbol{\theta}$, the analytic signed distance function of the boundary is computed at each cell corner, building a signed-distance field on the whole grid. We then compute a hyperplane of best fit (line in 2D; plane in 3D) to approximate the geometry of the boundary within each individual cell (where applicable). This hyperplane is used to compute the stiffness matrix component $\boldsymbol{K}^{ij}$ for the cell with indexing (i, j). Further, we use it to integrate the shape function $\mathcal{N}_i$ and the boundary condition $\boldsymbol{\alpha}$ described in Sec. 5 to form the linear constraints $\boldsymbol{Cv} = \boldsymbol{d}$. With the full QP formulated, $\tilde{\boldsymbol{v}}$ is obtained by solving the KKT conditions.

Signed-Distance Functions (SDF) for parametric shapes. Given design parameters $\boldsymbol{\theta}$ that determine the fluid-solid boundary, we first compute the signed distance from each cell's corners to the boundary. With the type of parametric surface known beforehand, evaluating this distance typically involves nothing more than analyzing the functions describing the level-set of the parametric shape. For example, if $\boldsymbol{\theta}$ parameterizes a circle with $\boldsymbol{\theta} = (\boldsymbol{c}, r)$ (the center position and radius of the circle), then the signed distance function can be written compactly as $\phi(\boldsymbol{x}) = r - \|\boldsymbol{c} - \boldsymbol{x}\|_2$ for any $\boldsymbol{x}$. Signed-distance functions of more sophisticated parametric shapes, *e.g.*, Bézier curves, can be found in our supplemental material. Throughout this paper, we use the convention that $\phi(\boldsymbol{x}) > 0$ refers to the solid region and $\phi(\boldsymbol{x}) < 0$ corresponds to the fluid region.

Boundary in a cell. Once the signed distances from a cell's corners to the boundary are given, the next step is to fit a hyperplane (line in 2D and plane in 3D) that approximates the fluid-solid interface in the cell when necessary. Note that this implicitly assumes the boundary does not contain delicate structures that are significantly smaller than the cell size. Depending on the signs at each corner, a cell is classified into three categories: purely in the interior of the solid region, purely in the interior of the fluid region, or partially in both regions. Only this final case requires fitting a hyperplane inside the cell, for which we obtain the hyperplane parameters from a linear least squares regression on the signed distances at its corners. As techniques for solving linear least squares are mature, we leave the details in our supplemental material.

At the end of this step, we have determined the type of each cell, and, for mixed-cells, we have provided an analytic means of approximating the boundary with a hyperplane. Such hyperplanes are crucial in assembling the matrices and vectors in Eqns. (27, 28).

Assembling $\boldsymbol{K}$. Computing $\boldsymbol{K}$ requires reasoning about two different types of cells: fluid cells and mixed cells. In the former case, the procedure for computing the contribution to $\boldsymbol{K}$ is well-established in the linear elasticity literature [Bendsoe and Sigmund 2013], as the integrals involved are evaluated over entire cells, either analytically or via Gauss quadrature rules. In the latter, mixed-cell case, it can be seen from Eqn. (24) that dependence on $\boldsymbol{\theta}$ only occurs via the area term. This is because the energy density function is evaluated at the same quadrature locations regardless of the cell type. The area function describes the ratio of the cell that is fluid and can be computed compactly with a single, closed-form expression [Barrow and Smith 1979] using the hyperplanes computed before:

Proposition 6.1. *Consider a d-dimensional halfspace $H = \{\boldsymbol{x} | \boldsymbol{a} \cdot \boldsymbol{x} + b \geq 0\}$ with the assumption that $\Pi_i a_i \neq 0$. The volume of its intersection with a unit hypercube is*

$$|[0,1]^d \cap H| = \sum_{\boldsymbol{q} \in F^0 \cap H} \frac{(-1)^{|\boldsymbol{q}_0|}(\boldsymbol{a} \cdot \boldsymbol{q} + b)^d}{d! \Pi_i a_i} \tag{29}$$

where $F^0 = \{0,1\}^d$ is the set of all hypercube corners and $|\boldsymbol{q}_0|$ is the number of zeros in the entries of $\boldsymbol{q}$.

Since this solution is closed-form and analytical, gradients can be simply and easily computed, which is especially beneficial in 3D, where plane-cube intersections can lead to complex cell fluid geometries.

Assembling $\boldsymbol{C}$ and $\boldsymbol{d}$. The remaining step is to compute $\boldsymbol{C}$ and $\boldsymbol{d}$ as a function of $\boldsymbol{\theta}$. We do this on a row-wise basis, again focusing on the (nontrivial) mixed cells. As established in Sec. 5, computing $\boldsymbol{C}$ and $\boldsymbol{d}$ requires computing the integral of the shape function $\mathcal{N}_i$ over the cross-sectional area of the boundary and the cell. Computing this integral analytically is tedious particularly in 3D because the cross-sectional area can have anywhere from three to six edges, and the situation would become even worse when computing the gradients. Thus (and keeping our procedure general), we design the following quadrature rule to approximate this line or area integral. Beginning with the quadrature points $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{2^d-1}$ which are in the centers of the cell quadrants (octants in 3D), we first project $\boldsymbol{x}_i$ onto the fluid-solid boundary approximated by our hyperplanes. In order to integrate a function over the boundary in the cell, we use these projections as the quadrature points to approximate the integral in Eqn. (25):

$$\int_{C_k \cap \Gamma_D} \mathcal{N}_i(\boldsymbol{x}) d\boldsymbol{x} \approx \sum_{j=0}^{2^d-1} \text{Proj}(\boldsymbol{x}_j; \boldsymbol{\theta}) \text{Area}(C_k^{(j)} \cap \partial\Omega) \tag{30}$$

where $\text{Proj}(\cdot; \boldsymbol{\theta})$ is an operator that projects a point onto the hyperplane and its reliance on $\boldsymbol{\theta}$ is due to the hyperplane parameters. The area function evaluates the cross-sectional area of the fluid-solid boundary and the quadrants or octants $C_k^{(j)}$. We calculate this area factor by noticing it is the directional derivative of $\text{Area}(C_k^{(j)} \cap \Omega)$ along the hyperplane's normal:

$$\text{Area}(C_k^{(j)} \cap \partial\Omega) = \frac{\partial \text{Area}(C_k^{(j)} \cap \Omega)}{\partial b} \|\boldsymbol{a}\|_2 \tag{31}$$

where $\boldsymbol{a}$ and b are defined as in Prop. 6.1. Computing this directional derivative requires nothing more than directly applying the chain rule to Prop 6.1, and we leave its details in our supplemental material.

Solving the QP problem. Having assembled every piece of Eqns. (27, 28), we demonstrate how to compute $\tilde{\boldsymbol{v}}$ as a function of $\boldsymbol{\theta}$. Since $\boldsymbol{K}$ (analogous to an elastic stiffness matrix) is positive semi-definite, the quadratic term is guaranteed to be convex. Thus, a (global) minimum for $\tilde{\boldsymbol{v}}$ always exists. We solve this QP by solving the KKT system:

$$\begin{pmatrix} \boldsymbol{K}(\boldsymbol{\theta}) & \boldsymbol{C}^T(\boldsymbol{\theta}) \\ \boldsymbol{C}(\boldsymbol{\theta}) & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \tilde{\boldsymbol{v}} \\ \tilde{\boldsymbol{\lambda}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{d}(\boldsymbol{\theta}) \end{pmatrix} \tag{32}$$

where $\tilde{\boldsymbol{\lambda}}$ is a Lagrange multiplier. We choose to solve this KKT system with a direct factorization method rather than apply an iterative optimization algorithm, as the pre-factorization of this system can help accelerate the gradient computation in the next section. Thus, solving $\tilde{\boldsymbol{v}}$ reduces to solving a symmetric (possibly indefinite) linear system depending purely on $\boldsymbol{\theta}$.

7 OPTIMIZATION

Given design parameters $\boldsymbol{\theta}$, we have described how to perform *forward* simulation in order to compute the steady-state velocity field $\tilde{\boldsymbol{v}}$. We now detail how to compute the *backward* gradient computation, *i.e.* computing the derivative of the loss function with respect to $\boldsymbol{\theta}$. We begin this section by first defining the loss function over which we wish to optimize and providing a means of evaluating the flow generated by simulation. Second, we discuss how gradients are computed via a scheme that back-propagates through the simulation. We conclude with a description of the full optimization algorithm.

Loss functions. While our method imposes no restrictions on the loss function as long as its gradients are well defined, we focus on a specific family of loss functions that penalize the discrepancy between the desired and actual velocity fields $\tilde{\boldsymbol{v}}$:

$$L(\tilde{\boldsymbol{v}}) = \|F(\tilde{\boldsymbol{v}}) - F(\boldsymbol{v}^*)\|_p \tag{33}$$

where $\boldsymbol{v}^*$ is a target velocity field, F is a function that extracts features we are interested in optimizing from a velocity field, and $\|\cdot\|_p$ is the p-norm. The choice of F is flexible and problem-specific. For example, F can be a selector function that picks velocities at the outlet of the device only, or F can be a curl operator for tasks focusing on optimizing the rotational speed of a velocity field.

Gradient computation. Given a complete description of the forward simulation scheme, deriving the gradients – at a high level – requires no more than iterative application of the chain rule. Each step in forward simulation has a corresponding step in the gradient computation that is then chained together. Most of these steps require the straightforward computation of the gradient of the output

Table 1. *A summary of our design problems. The "Time (s)/Function call" column reports the average wall-clock time of one function call to compute forward simulation and backpropagation. The time was measured on a single Intel(R) Xeon(R) CPU E7-4830 v4 @ 2.00GHz. The "Final loss" column reports the loss of our optimized design. The loss functions in all problems are normalized such that a unit loss refers to the average performance of randomly sampled designs and zero loss means an oracle design that perfectly matches the desired target, which may not exist in some problems.*

Name	Grid resolution	# Parameters	Level set	Boundary condition	Time (s)/Function call	Final loss
Amplifier	64×48	5	Béizer curves	No-separation	0.2	1.7e-5
Flow Averager	$64 \times 64 \times 4$	8	Béizer curves	No-separation	5.3	3.1e-2
Funnel	$64 \times 64 \times 16$	10	Béizer curves	No-separation	64.8	2.3e-1
Superposition Gate	$64 \times 64 \times 4$	5	Béizer curves	No-separation	4.9	4.9e-1
Fluidic Twister	$64 \times 64 \times 32$	32	NURBS surface	No-separation/No-slip	56.1	4.9e-2/9.7e-1
Fluidic Switch	$64 \times 64 \times 32$	26	NURBS surface	No-slip	171.1	5.8e-1

of the forward simulation (of that step) with respect to the input (of that step). Therefore, we leave the details of gradient computation in our supplemental material and only highlight one key step: the gradients of the solution of the QP problem. Specifically, we describe the computation of $\partial\tilde{\boldsymbol{v}}/\partial\boldsymbol{K}$, $\partial\tilde{\boldsymbol{v}}/\partial\boldsymbol{C}$, and $\partial\tilde{\boldsymbol{v}}/\partial\boldsymbol{d}$. In order to avoid unwieldy, high-dimensional tensor notation, we describe the gradient derivation in differential form (sufficient for the purpose of computing the gradients). Concretely, given perturbations $\delta\boldsymbol{K}$, $\delta\boldsymbol{C}$, and $\delta\boldsymbol{d}$, we explain how much perturbation $\delta\tilde{\boldsymbol{v}}$ is expected.

Differentiation the KKT system in Eqn. (32) results in the following linear system with $\delta\tilde{\boldsymbol{v}}$ and $\delta\tilde{\boldsymbol{\lambda}}$ as unknowns (we omit the $\boldsymbol{\theta}$ dependence for clarity):

$$\begin{pmatrix} \boldsymbol{K} & \boldsymbol{C}^T \\ \boldsymbol{C} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \delta\tilde{\boldsymbol{v}} \\ \delta\tilde{\boldsymbol{\lambda}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{0} \\ \delta\boldsymbol{d} \end{pmatrix} - \begin{pmatrix} \delta\boldsymbol{K} & \delta\boldsymbol{C}^T \\ \delta\boldsymbol{C} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \tilde{\boldsymbol{v}} \\ \tilde{\boldsymbol{\lambda}} \end{pmatrix} \tag{34}$$

$\delta\boldsymbol{K}$, $\delta\boldsymbol{C}$, and $\delta\boldsymbol{d}$ are all known; $\tilde{\boldsymbol{v}}$ and $\tilde{\boldsymbol{\lambda}}$ have been computed during the forward simulation process. Since the linear system of equations here has precisely the same left-hand side as the KKT system solved in the forward simulation, this matrix can be pre-factorized once during simulation and reused during gradient computation, allowing for efficient solving of $\delta\tilde{\boldsymbol{v}}$.

Optimization. With a method for computing gradients of the loss backward through simulation with respect to design parameters in tow, we are able to apply gradient-based, quasi-Newton methods for efficient optimization. The performance of this approach primarily depends on two crucial factors: the specific local optimization method chosen, and the initial guess. Since all of our design problems have nonlinear continuous losses and bound constraints on parameters only, we chose L-BFGS, a classic quasi-Newton method, as our optimizer. The initial guess was selected by picking the best design among a number of randomly sampled designs in the design space. Sampling designs serves two purposes in our method: first, it reduces the risk of getting trapped into local minima. Second, we can rescale the loss function in each design problem by setting the average loss from these randomly samples as the unit loss. After this normalization, the loss functions from different design problems share the same physical meaning: loss = 0 means an oracle solution that matches the target perfectly, which is not always possible, and loss = 1 means the solution has an empirically average performance among all possible designs.

8 RESULTS AND DISCUSSIONS

In this section, we present six 2D and 3D design problems to evaluate the performance of our implementation of the differentiable Stokes flow as well as the optimization pipeline. We start this section by describing the problem statements for each design problem, followed by evaluations and discussions on the experimental results. We ask readers to refer to our supplemental video for a complete demonstration of these design problems, the evolution of our optimization process, and the animation of our final results.

8.1 Design Problems

We summarize the basic information about these design problems in Table 1 and Fig. 1, 4, and 5. The number of decision variables in these design problems ranges from 5 to 32, and the cell resolution of the scenes varies from 64×48 cells in 2D to $64 \times 64 \times 32$ cells in 3D. Below we discuss the setup of each design problem in detail:

Amplifier. This motivating example in Fig. 2 is a 2D design problem that aims to amplify the velocity of inlet flow by a factor of 3. The design variables are the control points of the Bézier curves representing the upper and lower solid-fluid boundaries. The inlet flow enters the domain from the left with a velocity of $(v, 0)$, and the loss function is defined as the difference between $(3v, 0)$ and the average speed of the outlet flow on the right.

Flow Averager. The goal of this design problem is to engineer a fluidic load balancer with two inputs (left) and two outputs (right). Let the two inlet flows enter the domain with velocities $(v_{i_1}, 0, 0)$ and $(v_{i_2}, 0, 0)$ where v_{i_1} and v_{i_2} are arbitrary numbers and let $\boldsymbol{v}_{o_1}$ and $\boldsymbol{v}_{o_2}$ be the average flow velocities at the two outlets. The objective is to encourage both $\boldsymbol{v}_{o_1}$ and $\boldsymbol{v}_{o_2}$ to be as close as possible to $((v_{i_1} + v_{i_2})/2, 0, 0)$ (Fig. 4 top). In other words, we expect the design to average the two inputs no matter what values are given for v_{i_1} and v_{i_2}. We optimize a loss function that concurrently optimizes for these two basis inputs $(v_{i_1}, v_{i_2}) = (0, 1)$ and $(1, 0)$. The design space consists of four Bézier curves in 2D. The 3D solid-fluid boundaries are formed by extruding these curves vertically.

Funnel. This design problem considers a fluidic domain with two inputs and one output. The goal is to design the internal boundary of the fluidic domain such that the direction of the output flow is 45-degrees and input-invariant (Fig. 4 middle). Let $(v_{i_1}, 0, 0)$ and

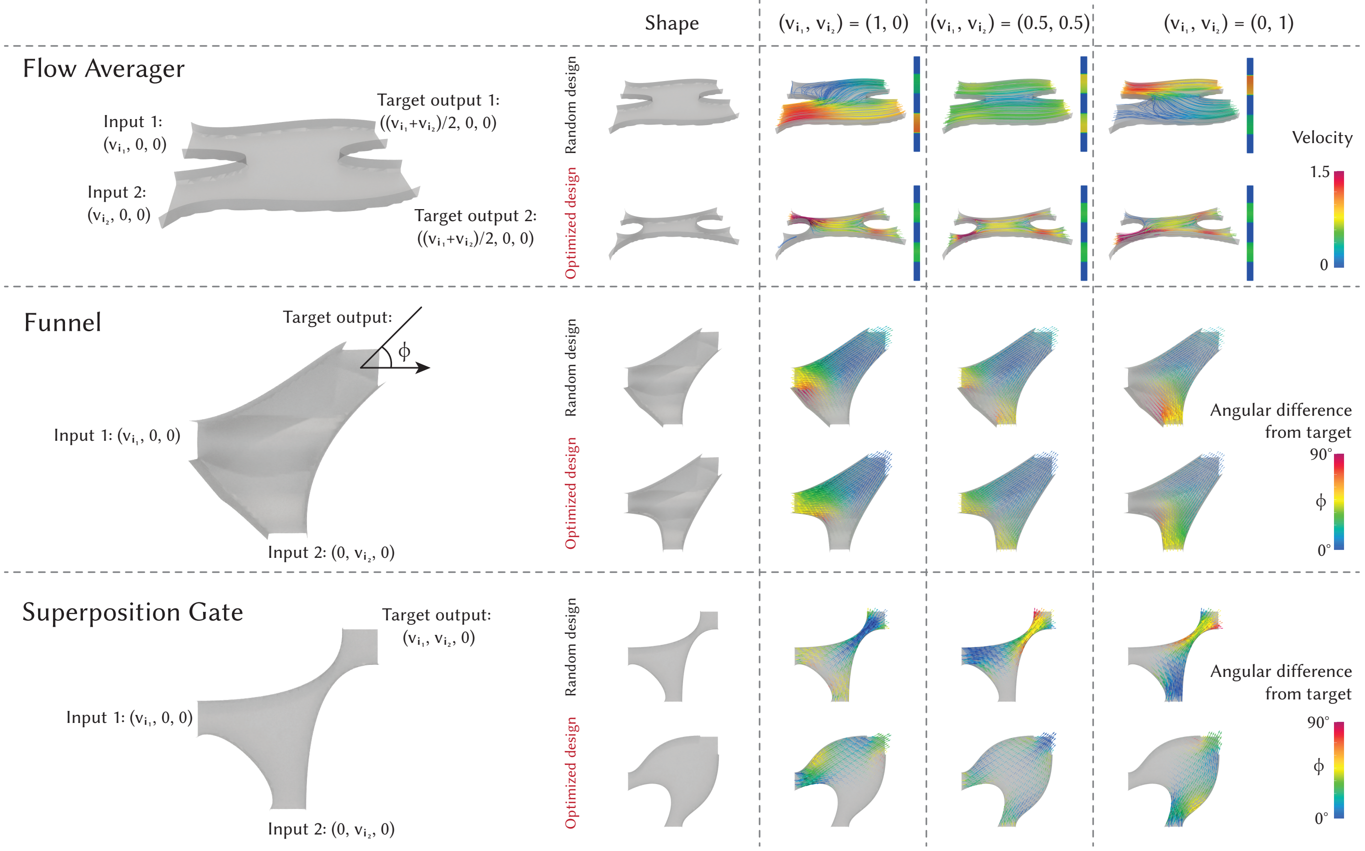
Shape
$(v_{i_1}, v_{i_2}) = (1, 0)$
$(v_{i_1}, v_{i_2}) = (0.5, 0.5)$
$(v_{i_1}, v_{i_2}) = (0, 1)$
Flow Averager
Input 1: $(v_{i_1}, 0, 0)$
Input 2: $(v_{i_2}, 0, 0)$
Target output 1: $((v_{i_1}+v_{i_2})/2, 0, 0)$
Target output 2: $((v_{i_1}+v_{i_2})/2, 0, 0)$
Random design
Optimized design
Velocity
1.5
0
Funnel
Target output:
ϕ
Input 1: $(v_{i_1}, 0, 0)$
Input 2: $(0, v_{i_2}, 0)$
Angular difference from target
90°
ϕ
0°
Superposition Gate
Target output: $(v_{i_1}, v_{i_2}, 0)$
Input 1: $(v_{i_1}, 0, 0)$
Input 2: $(0, v_{i_2}, 0)$

Fig. 4. *Three optimization examples: the Flow Averager, the Funnel, and the Superposition Gate. The left figure of each example shows the specifications of the design problem. The right eight figures of each example show the comparison between a randomly sampled design (top row) and the optimized design (bottom row) with three different inputs. In the Flow Averager, the vertical color bar inside each inset indicates the velocity magnitude at the cross section of the two outlets, and solutions with two outlets having more similar colors are better. In the Funnel and the Superposition Gate, the color indicates the angular error between the local velocity and the target velocity (cooler at the outlet is better).*

$(0, v_{i_2}, 0)$ be the two inlet flow velocities; the design is evaluated by continuously varying the inputs from $(v_{i_1}, v_{i_2}) = (1, 0)$ and $(0, 1)$ and observing the change in the direction of the outlet flow. Note that while the design space consists of 2D Béizer curves only, the bumpy bottom plate creates enough vertical variations to make it our first 3D design problem.

Superposition Gate. This design problem shares the similar setting with the Funnel above except that the goal is to obtain an outlet flow with a velocity of $(v_{i_1}, v_{i_2}, 0)$ (Fig. 4 bottom); thus the name superposition gate. When the inputs (v_{i_1}, v_{i_2}) vary from $(1, 0)$ to $(0, 1)$, an ideal superposition gate design should generate an outlet flow that sweeps the first quadrant.

Fluidic Twister. This 3D problem considers designing the internal surface of a tunnel to generate a twisted flow (Fig. 5). With a straight inlet flow $\boldsymbol{v}_i = (0, 0, -1)$ entering the tunnel from the top, an ideal design needs to generate an outlet velocity field $\boldsymbol{v}_o = (u_o, v_o, w_o)$ at the bottom such that it has a desired vertical curl ω: $\nabla \times (u_o, v_o, 0) = (0, 0, 2\omega)$. We discretize the curl operator on our grid and define the loss function as the difference between $\nabla \times (u_o, v_o, 0)$ and $(0, 0, 2\omega)$. The internal surface of the tunnel is represented by a NURBS surface with 32 free control points to be optimized.

Fluidic Switch. In this problem, we consider a fluidic device with a switch that can kinematically change the fluid-solid boundary in a fluidic domain. By switching on and off, we expect the outlet flow velocity from the device to transition between two prescribed velocity profiles (Fig. 1). We set up our fluidic domain with one input, two outputs, and a solid obstacle immersed in the fluidic domain. The solid obstacle is parameterized by a NURBS surface whose 24 control points are to be optimized, and it is pinned on the bottom plate so it can rotate along a vertical axis. The two states of the switch set the rotational angle of the solid obstacle to two different values, and the loss function equals the sum of the losses from the two states, which is defined as the difference between the actual output velocity and a prescribed desired velocity profile (Fig. 1 rightmost). Note that the switching angle in this design problem is also a parameter to be optimized.

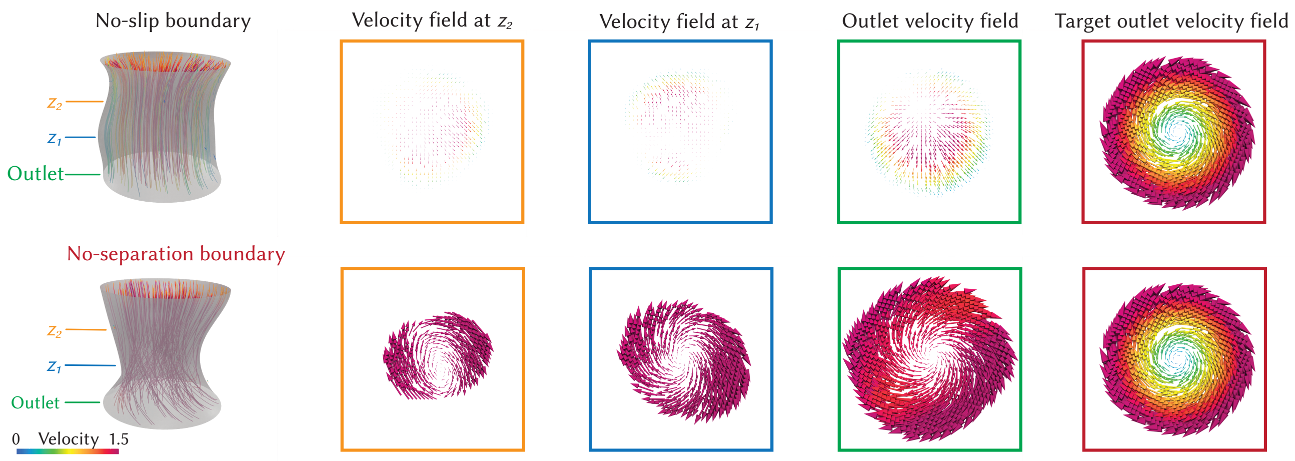

Fig. 5. *A comparison between optimizing the Fluidic Twister with no-slip boundaries (top) and no-separation boundaries (bottom). We show the velocity field at three cross-sectional areas (middle three columns) of the optimized design (left column) as well as the target, twisted field (right column) at the outlet (green). With the no-slip boundary, the resulting velocity field is attenuated significantly. With the no-separation boundary, a desired helical pattern emerges to facilitate the swirling of the outlet flow.*

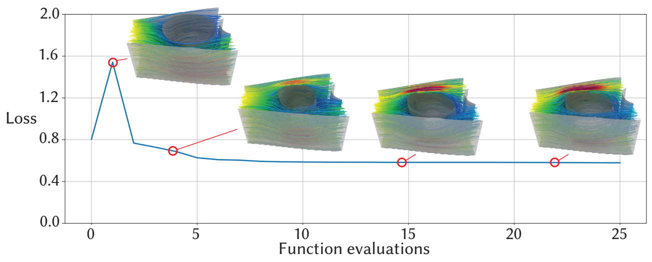

Fig. 6. *The evolution of the shape and the loss for the design of the Fluidic Switch over 25 function calls.*

8.2 Evaluation

Implementation details. We implemented our algorithm in C++ on a Linux workstation with 8 CPU cores and 32G memory. We used PARDISO [Schenk and Gärtner 2004], a parallel linear system solver for symmetric indefinite matrices, to solve the linear systems of equations in both forward simulation and gradient computation. To speed up the computation, during each function call to compute forward simulation, we pre-factorized the matrix and reused the factorization in gradient computation with new right-hand sides for gradient computation. For each design problem, we start our optimization by sampling random designs and picking the best one to initialize the L-BFGS local optimization algorithm. The actual number of samples depends on the complexity of the design problem. In our experiments, we used 10 samples for problems with $\leq$ 10 parameters and 100 samples for larger problems. Random sampling does not create a significant time burden for our algorithm because it is easily parallelizable and requires forward simulation only. We terminated the optimizer when a maximum number of function evaluations (50 in our experiments) was reached or the solution converged into a local minimum. For all examples, we consistently observed their convergence before the maximum number of function evaluations is reached.

Performance improvement. We report the statistics about the optimization process in Table 1 and the final designs discovered by our algorithm in Fig. 1, 2, 4, 5, and 6. Our algorithm improved the initial guess across the board and the final design performed significantly better than an average design (loss = 1) in all examples, with the actual improvement margin largely depending on the complexity of each problem.

It is worth mentioning that many of the novel designs revealed by our algorithm not only are physically plausible but also match the physical intuition behind the design intent. For example, in the Amplifier problem, the evolution of the boundaries narrowed the outlet so that the flow jets at a desired, faster speed (Fig. 2). In the Funnel example, a diagonal tunnel was formed near the end of the output in order to enforce an outlet flow whose direction is invariant to inputs (Fig. 4 middle). The most notable discovery of the novel design comes from the Fluidic Twister: although the internal surface is parameterized by a NURBS surface, the final solution strongly resembles a helical surface generated by a rotated, descending ellipsoid (Fig. 5). The emergence of a helical surface in this design problem is no coincidence and clearly reflects the design intent of generating a swirling flow.

Linearity in the fluidic devices. The KKT system in Sec. 6 connects the velocity field $\boldsymbol{v}$, the right-hand side of all boundary conditions, and the design parameters $\boldsymbol{\theta}$ in a single linear system of equations whose left-hand matrix depends on $\boldsymbol{\theta}$ only. As a result, when we fix $\boldsymbol{\theta}$, the response of the system is a linear function of the input to the system, which comes naturally from the analogy between Stokes flow and linear elasticity. Moreover, since by definition Stokes flow is steady-state, the fluidic system we investigate in this paper is therefore linear time-invariant (LTI). It is well known that the behavior of an LTI system can be fully analyzed and well understood by investigating its response to a small number of base inputs, and we made heavy use of this fact in our experiments to simplify the design problem. For example, when designing the Funnel, it is sufficient to ensure the outlet flow is diagonal under two inputs $(v_{i_1}, v_{i_2}) = (1, 0)$ and $(0, 1)$ only, and the outlet flow in response to $(0.5, 0.5)$ equals the average of the outlet flows from inputs $(1, 0)$ and $(0, 1)$ (Fig. 4 middle).

It is worth pointing out that while the fluidic system is LTI with a *fixed* $\boldsymbol{\theta}$, the full optimization problem is still highly nonlinear and non-convex. This is because $\boldsymbol{\theta}$ parameterizes the fluid-solid boundary in a nontrivial way, which is embedded in the left-hand side of the KKT system.

Boundary conditions. Our convenient, explicit boundary conditions are flexible, and are a key ingredient in unlocking many of the demonstrations here. Particularly of note are the Fluidic Twister and the Fluidic Gate examples. No-separation boundaries are necessary to build up the circumferential "swirling" motion seen in the Twister example. The no-slip boundary condition, on the other hand, significantly dampens the velocities along the fluid-solid boundary, inhibiting the vortical flow. The two boundary conditions led to significantly different optimization results (Table 1): while a Twister optimized with a no-separation boundary achieves almost optimal performance (loss ≈ 0), optimization with a no-slip boundary hardly made any progress and performs no better than a random guess (loss ≈ 1). By contrast, the Fluidic Switch relies on no-slip boundary conditions to dampen the flow along the "off" path. If no-separation boundary conditions were to be used here, the rotational switch would need to be physically translated between configurations to completely block the "off" branches to achieve zero velocity; otherwise, some non-zero velocity will persist. We stress that our accommodation of several boundary conditions is a feature, as an engineer can achieve any of them by selecting appropriate materials along the boundary interface.

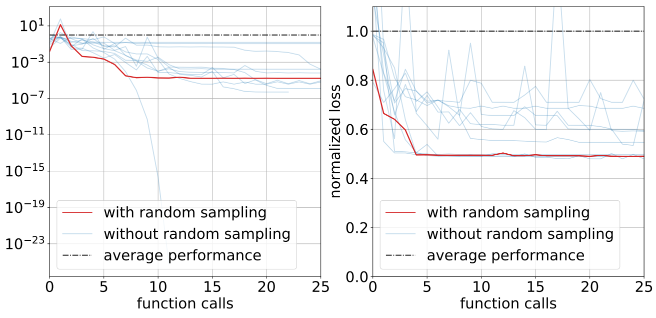

Fig. 7. *Ablation study on the necessity of global random sampling in two design problems: Amplifier (left) and Superposition Gate (right). Each blue line indicates the process of running L-BFGS optimization directly from a random design, and the red line shows our optimization progress with an initial guess from the best of 10 random designs. While a particularly good random seed can outperform our method, the flat tails from multiple random seeds reveal that local minima are common in such design problems.*

Local minima. Since our gradient-based optimization pipeline is inherently a local optimization method, it can suffer from getting trapped into local minima (Fig. 7). The distribution of local minima is problem-specific and affected heavily by the landscape of the loss function. We have partially alleviated this issue by randomly sampling multiple guesses prior to optimization and picking the best as an initial guess. While more advanced global search heuristics can be applied to our pipeline, we found this simple random sampling scheme sufficient to generate reasonably functional devices in all our design problems.

Solution convergence. To prove our simulation converges under refinement and verify our governing equations approximate the truly incompressible Stokes flow, we experimented with the 2D amplifier example with an initial resolution of 32×24 cells. We then subdivided the domain by a factor of 2, 4, 8, 16, and 32, resulting in a resolution of 1024×768 eventually (Fig. 8 top). Additionally, we started with 32×24 cells and increased the Poisson's ratio from 0.45 until 0.499 (Fig. 8 bottom). We observed that in both cases, the velocity fields converged to a limit, which indicates that our quasi-incompressible Stokes flow model well approximates the truly incompressible Stokes flow.

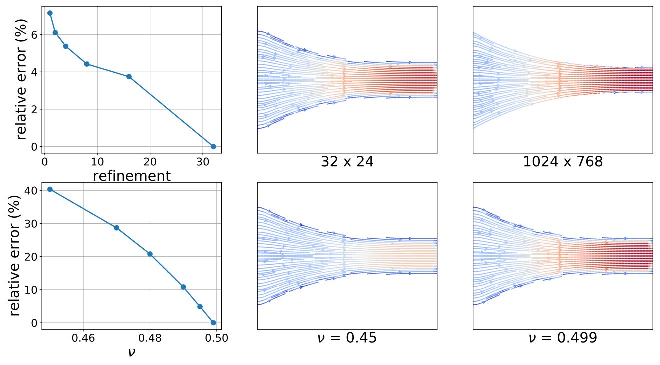

Fig. 8. *Convergence of our quasi-incompressible Stokes flow tested on the Amplifier example. Top: we solve the velocity field starting with* 32×24 *cells (middle) and increase the resolution by a factor of* 2, 4, 8, 16, *and* 32 *(right). The relative error (left, measured by comparing to the solution solved with* 32× *resolution) decreases as the velocity field is solved under refinement. Bottom: we solve the velocity field with* $\nu = 0.45$ *(middle), 0.47, 0.48, 0.49, 0.495, and 0.499 (right). The relative error (left, measured by comparing to* $\nu = 0.499$*) converges to 0 as* ν *converges to 0.5.*

9 LIMITATIONS AND FUTURE WORK

The long-term vision of computational fluidics design is ambitious, ultimately aspiring to the automated design of complex devices such as engines, pumps, and heart valves. Optimizing such machinery, however, is extremely challenging, requiring modeling of complex fluid dynamics while optimizing over highly complex components. We see our work as a meaningful first step toward this eventual goal. We took the Stokes flow as our fluid model, which has been used widely in engineering design and optimization problems over the past decades to model the steady-state, linearized fluidic transportation problems with a relatively low Reynolds number. Further, Stokes flow is computationally well-suited to design problems, as it is well-conditioned, linear, and provides smooth gradients, allowing for a fast inner loop of complex outer design problems. An interesting future direction to explore is to improve the expressibility of the fluid simulation method. Particularly interesting would be a steady-state Navier-Stokes fluid simulator that considers the effect of an advection term. It is also interesting to consider the effect of deformable boundaries, allowing for the design of devices with fluid-elastic coupling for applications in, e.g., soft robotics.

The second drawback of our method lies in our choice of parameterized level-sets as a design space. Such a design space was deliberately chosen as it allows for sub-grid shape design with smooth, clearly defined boundaries that separate fluid and solid regions, a common failing of topology optimization, which provides no such guarantees and only operates on the non-smooth grid cells themselves (thus making boundary conditions tricky to reason about). Still, this parameterization must be chosen by a user. A tractable method for searching both over topology while keeping boundaries smooth and regular is desired.

Third, although our Stokes solver allows for sub-grid resolution, it does not scale to arbitrarily large scenes, as it is bottlenecked by the performance of our choice of linear system solver and the optimizer. A parallel multigrid solver along with application of the adjoint method in gradient computation would allow our framework to scale to support larger problems. It would also be interesting to explore other optimizers like alternating direction method of multipliers (ADMM) [Overby et al. 2017] in our problem especially when the objective is separable on variables.

Fourth, although we purposefully kept our simulation physics-based so as to make our method amenable to real-world manufacturing, we did not fabricate and test any of our devices. Our parameterization allows engineers to specify boundaries that are physically manufacturable, without the worry for non-manufacturable parts (such as disconnected pieces in 3D). It would be interesting to physically fabricate our optimized devices and benchmark the predictive accuracy of the simulation as compared to the realized flow.

Finally, despite our initial sampling pre-processing step, whose coarse global search improves over a random starting point, there is no guarantee that our algorithm will converge to a global minimum. This is a drawback of all local continuous optimization methods like the one we employ. While smoothness of our domain helps in that we rarely find *bad* local minima, an algorithm for finding more globally optimal solutions is desired.

ACKNOWLEDGMENTS

Tao Du, Kui Wu, and Andrew Spielberg would like to thank Buttercup Foshey (and also Michael Foshey) for moral support during this work. Wojciech Matusik acknowledges the funding support from IARPA under grant 2019-19020100001. Bo Zhu acknowledges the funding supports from Neukom Institute CompX Faculty Grant, Burke Research Initiation Award, NSF 1919647, and Toyota TEMA North America Inc. Eftychios Sifakis acknowledges the funding supports from NSF IIS-2008584, CCF-1812944, and IIS-1763638.

REFERENCES

Niels Aage, Erik Andreassen, Boyan S. Lazarov, and Ole Sigmund. 2017. Giga-Voxel Computational Morphogenesis for Structural Design. *Nature* 550, 7674 (2017), 84–86.

Niels Aage, Thomas H. Poulsen, Allan Gersborg-Hansen, and Ole Sigmund. 2008. Topology Optimization of Large Scale Stokes Flow Problems. *Structural and Multidisciplinary Optimization* 35, 2 (2008), 175–180.

Joe Alexandersen and Casper Schousboe Andreasen. 2020. A Review of Topology Optimisation for Fluid-Based Problems. *Fluids* 5, 1 (2020), 29.

Ryoichi Ando, Nils Thürey, and Chris Wojtan. 2013. Highly Adaptive Liquid Simulations on Tetrahedral Meshes. *ACM Trans. Graph.* 32, 4, Article 103 (July 2013), 10 pages. https://doi.org/10.1145/2461912.2461982

Casper Schousboe Andreasen and Ole Sigmund. 2013. Topology Optimization of Fluid-Structure-Interaction Problems in Poroelasticity. *Computer Methods in Applied Mechanics and Engineering* 258 (2013), 55–62.

Vinicius C. Azevedo, Christopher Batty, and Manuel M. Oliveira. 2016. Preserving Geometry and Topology for Fluid Flows with Thin Obstacles and Narrow Gaps. *ACM Trans. Graph.* 35, 4, Article 97 (July 2016), 12 pages. https://doi.org/10.1145/2897824.2925919

David L. Barrow and Philip W. Smith. 1979. Spline Notation Applied to a Volume Problem. *The American Mathematical Monthly* 86, 1 (1979), 50–51.

Christopher Batty, Florence Bertails, and Robert Bridson. 2007. A Fast Variational Framework for Accurate Solid-Fluid Coupling. In *ACM SIGGRAPH 2007 Papers* (San Diego, California) *(SIGGRAPH '07)*. Association for Computing Machinery, New York, NY, USA, 100–es. https://doi.org/10.1145/1275808.1276502

William Baxter, Yuanxin Liu, and Ming C. Lin. 2004. A Viscous Paint Model for Interactive Applications. *Computer Animation and Virtual Worlds* 15, 3-4 (2004), 433–441.

Jacob Bedrossian, James H. Von Brecht, Siwei Zhu, Eftychios Sifakis, and Joseph Teran. 2010. A Second Order Virtual Node Method for Elliptic Problems with Interfaces and Irregular Domains. *J. Comput. Phys.* 229, 18 (2010), 6405–6426.

Reza Behrou, Ram Ranjan, and James K. Guest. 2019. Adaptive Topology Optimization for Incompressible Laminar Flow Problems with Mass Flow Constraints. *Computer Methods in Applied Mechanics and Engineering* 346 (2019), 612–641.

Martin Philip Bendsoe and Ole Sigmund. 2013. *Topology Optimization: Theory, Methods, and Applications.* Springer Science & Business Media.

Haimasree Bhattacharya, Michael Bang Nielsen, and Robert Bridson. 2012. Steady State Stokes Flow Interpolation for Fluid Control. In *Eurographics (Short Papers)*. Citeseer, 57–60.

Thomas Borrvall and Joakim Petersson. 2003. Topology Optimization of Fluids in Stokes Flow. *International Journal for Numerical Methods in Fluids* 41, 1 (2003), 77–107.

Franco Brezzi and Michel Fortin. 2012. *Mixed and Hybrid Finite Element Methods.* Vol. 15. Springer Science & Business Media.

Robert Bridson. 2015. *Fluid Simulation for Computer Graphics.* CRC press.

Walter Jesus Paucar Casas and Renato Pavanello. 2017. Optimization of Fluid-Structure Systems by Eigenvalues Gap Separation with Sensitivity Analysis. *Applied Mathematical Modelling* 42 (2017), 269–289.

Vivien J. Challis and James K. Guest. 2009. Level Set Topology Optimization of Fluids in Stokes Flow. *Internat. J. Numer. Methods Engrg.* 79, 10 (2009), 1284–1308.

Mengyu Chu and Nils Thürey. 2017. Data-Driven Synthesis of Smoke Flows with CNN-Based Feature Descriptors. *ACM Trans. Graph.* 36, 4, Article 69 (July 2017), 14 pages. https://doi.org/10.1145/3072959.3073643

Christophe Daux, Nicolas Moës, John Dolbow, Natarajan Sukumar, and Ted Belytschko. 2000. Arbitrary Branched and Intersecting Cracks with the Extended Finite Element Method. *Internat. J. Numer. Methods Engrg.* 48, 12 (2000), 1741–1760.

Joshua D. Deaton and Ramana V. Grandhi. 2014. A Survey of Structural and Multidisciplinary Continuum Topology Optimization: Post 2000. *Structural and Multidisciplinary Optimization* 49, 1 (2014), 1–38.

Cetin B. Dilgen, Sumer B. Dilgen, David R. Fuhrman, Ole Sigmund, and Boyan S. Lazarov. 2018a. Topology Optimization of Turbulent Flows. *Computer Methods in Applied Mechanics and Engineering* 331 (2018), 363–393.

Sumer B. Dilgen, Cetin B. Dilgen, David R. Fuhrman, Ole Sigmund, and Boyan S. Lazarov. 2018b. Density Based Topology Optimization of Turbulent Flow Heat Transfer Systems. *Structural and Multidisciplinary Optimization* 57, 5 (2018), 1905–1918.

Marie-Lena Eckert, Kiwon Um, and Nils Thürey. 2019. ScalarFlow: A Large-Scale Volumetric Data Set of Real-World Scalar Transport Flows for Computer Animation and Machine Learning. *ACM Trans. Graph.* 38, 6, Article 239 (Nov. 2019), 16 pages. https://doi.org/10.1145/3355089.3356545

Douglas Enright, Stephen Marschner, and Ronald Fedkiw. 2002. Animation and Rendering of Complex Water Surfaces. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques.* 736–744.

Anton Evgrafov. 2006. Topology Optimization of Slightly Compressible Fluids. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik: Applied Mathematics and Mechanics* 86, 1 (2006), 46–62.

Ronald Fedkiw, Tariq Aslam, Barry Merriman, Stanley Osher, et al. 1999. A Non-Oscillatory Eulerian Approach to Interfaces in Multimaterial Flows (the Ghost Fluid Method). *J. Comput. Phys.* 152, 2 (1999), 457–492.

Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. 2001. Visual Simulation of Smoke. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques.* 15–22.

Bryan E. Feldman, James F. O'Brien, Bryan M. Klingner, and Tolga G. Goktekin. 2005. Fluids in Deforming Meshes. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Los Angeles, California) *(SCA '05)*. Association for Computing Machinery, New York, NY, USA, 255–259. https://doi.org/10.1145/1073368.1073405

Florian Ferstl, Rüdiger Westermann, and Christian Dick. 2014. Large-Scale Liquid Simulation on Adaptive Hexahedral Grids. *IEEE Transactions on Visualization and Computer Graphics* 20, 10 (2014), 1405–1417.

Francisco. J. Gaspar, José L. Gracia, Francisco J. Lisbona, and Cornelis W. Oosterlee. 2008. Distributive Smoothers in Multigrid for Problems with Dominating Grad–Div Operators. *Numerical Linear Algebra with Applications* 15, 8 (2008), 661–683. https://doi.org/10.1002/nla.587 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/nla.587

Allan Gersborg-Hansen, Ole Sigmund, and Robert B. Haber. 2005. Topology Optimization of Channel Flow Problems. *Structural and Multidisciplinary Optimization* 30, 3 (2005), 181–192.

James K. Guest and Jean H. Prévost. 2006. Topology Optimization of Creeping Fluid Flows Using a Darcy–Stokes Finite Element. *Internat. J. Numer. Methods Engrg.* 66, 3 (2006), 461–484.

Jeffrey Lee Hellrung, Luming Wang, Eftychios Sifakis, and Joseph Teran. 2012. A Second Order Virtual Node Method for Elliptic Problems with Interfaces and Irregular Domains in Three Dimensions. *J. Comput. Phys.* 231, 4 (2012), 2015–2048.

Philipp Holl, Vladlen Koltun, and Nils Thürey. 2020. Learning to Control PDEs with Differentiable Physics. In *International Conference on Learning Representations.*

Yuanming Hu, Jiancheng Liu, Andrew Spielberg, Joshua B. Tenenbaum, William T. Freeman, Jiajun Wu, Daniela Rus, and Wojciech Matusik. 2019. ChainQueen: A Real-Time Differentiable Physical Simulator for Soft Robotics. In *International Conference on Robotics and Automation.* IEEE, 6265–6271.

Thomas J. R. Hughes. 2012. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis.* Courier Corporation.

Hikaru Ibayashi, Chris Wojtan, Nils Thürey, Takeo Igarashi, and Ryoichi Ando. 2018. Simulating Liquids on Dynamically Warping Grids. *IEEE Transactions on Visualization and Computer Graphics* (2018).

Byungsoo Kim, Vinicius C. Azevedo, Nils Thürey, Theodore Kim, Markus Gross, and Barbara Solenthaler. 2019. Deep Fluids: A Generative Network for Parameterized Fluid Simulations. In *Computer Graphics Forum*, Vol. 38. Wiley Online Library, 59–70.

Nipun Kwatra, Jonathan Su, Jón T. Grétarsson, and Ronald Fedkiw. 2009. A Method for Avoiding the Acoustic Time Step Restriction in Compressible Flow. *J. Comput. Phys.* 228, 11 (2009), 4146–4161.

Egor Larionov, Christopher Batty, and Robert Bridson. 2017. Variational Stokes: A Unified Pressure-Viscosity Solver for Accurate Viscous Liquids. *ACM Trans. Graph.* 36, 4, Article 101 (July 2017), 11 pages. https://doi.org/10.1145/3072959.3073628

Benny Lautrup. 2004. *Physics of Continuous Matter: Exotic and Everyday Phenomena in the Macroscopic World.* CRC press.

Sung-Hee Lee, Eftychios Sifakis, and Demetri Terzopoulos. 2009. Comprehensive Biomechanical Modeling and Simulation of the Upper Body. *ACM Trans. Graph.* 28, 4, Article 99 (Sept. 2009), 17 pages. https://doi.org/10.1145/1559755.1559756

Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B. Tenenbaum, and Antonio Torralba. 2019. Learning Particle Dynamics for Manipulating Rigid Bodies, Deformable Objects, and Fluids. (2019).

Junbang Liang, Ming C. Lin, and Vladlen Koltun. 2019. Differentiable Cloth Simulation for Inverse Problems. In *Advances in Neural Information Processing Systems.* 771–780.

Sen Lin, Longyu Zhao, James K. Guest, Timothy P. Weihs, and Zhenyu Liu. 2015. Topology Optimization of Fixed-Geometry Fluid Diodes. *Journal of Mechanical Design* 137, 8 (2015).

Haixiang Liu, Yuanming Hu, Bo Zhu, Wojciech Matusik, and Eftychios Sifakis. 2018. Narrow-Band Topology Optimization on a Sparsely Populated Grid. *ACM Trans. Graph.* 37, 6, Article 251 (Dec. 2018), 14 pages. https://doi.org/10.1145/3272127.3275012

Kurt Maute and Matthew Allen. 2004. Conceptual Design of Aeroelastic Structures by Topology Optimization. *Structural and Multidisciplinary Optimization* 27, 1-2 (2004), 27–42.

Aleka McAdams, Yongning Zhu, Andrew Selle, Mark Empey, Rasmus Tamstorf, Joseph Teran, and Eftychios Sifakis. 2011. Efficient Elasticity for Character Skinning with Contact and Collisions. *ACM Trans. Graph.* 30, 4, Article 37 (July 2011), 12 pages. https://doi.org/10.1145/2010324.1964932

Antoine McNamara, Adrien Treuille, Zoran Popović, and Jos Stam. 2004. Fluid Control Using the Adjoint Method. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 449–456. https://doi.org/10.1145/1015706.1015744

Nicolas Moës, John Dolbow, and Ted Belytschko. 1999. A Finite Element Method for Crack Growth without Remeshing. *Internat. J. Numer. Methods Engrg.* 46, 1 (1999), 131–150.

Maxim Olshanskii, Gert Lube, Timo Heister, and Johannes Löwe. 2009. Grad-Div Stabilization and Subgrid Pressure Models for the Incompressible Navier-Stokes Equations. *Computer Methods in Applied Mechanics and Engineering* 198, 49-52 (2009), 3975–3988.

Stanley Osher and Ronald Fedkiw. 2003. *Level Set Methods and Dynamic Implicit Surfaces.* Applied Mathematical Sciences, Vol. 153. Springer. I–XIII, 1–273 pages.

Matthew Overby, George E Brown, Jie Li, and Rahul Narain. 2017. ADMM $\supseteq$ Projective Dynamics: Fast Simulation of Hyperelastic Models with Dynamic Constraints. *IEEE Transactions on Visualization and Computer Graphics* 23, 10 (2017), 2222–2234.

Zherong Pan, Jin Huang, Yiying Tong, Changxi Zheng, and Hujun Bao. 2013. Interactive Localized Liquid Motion Editing. *ACM Trans. Graph.* 32, 6, Article 184 (Nov. 2013), 10 pages. https://doi.org/10.1145/2508363.2508429

Evangelos M. Papoutsis-Kiachagias and Kyriakos C. Giannakoglou. 2016. Continuous Adjoint Methods for Turbulent Flows, Applied to Shape and Topology Optimization: Industrial Applications. *Archives of Computational Methods in Engineering* 23, 2 (2016), 255–299.

Taylor Patterson, Nathan Mitchell, and Eftychios Sifakis. 2012. Simulation of Complex Nonlinear Elastic Bodies Using Lattice Deformers. *ACM Trans. Graph.* 31, 6, Article 197 (Nov. 2012), 10 pages. https://doi.org/10.1145/2366145.2366216

Karthik Raveendran, Nils Thürey, Chris Wojtan, and Greg Turk. 2012. Controlling Liquids Using Meshes. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Lausanne, Switzerland) *(SCA '12)*. Eurographics Association, Goslar, DEU, 255–264.

George I. N. Rozvany. 2009. A Critical Review of Established Methods of Structural Topology Optimization. *Structural and Multidisciplinary Optimization* 37, 3 (2009), 217–237.

Connor Schenck and Dieter Fox. 2018. SPNets: Differentiable Fluid Dynamics for Deep Neural Networks *(Proceedings of Machine Learning Research, Vol. 87)*. PMLR, 317–335. http://proceedings.mlr.press/v87/schenck18a.html

Olaf Schenk and Klaus Gärtner. 2004. Solving Unsymmetric Sparse Systems of Linear Equations with PARDISO. *Future Generation Computer Systems* 20, 3 (2004), 475–487.

Craig Schroeder, Wen Zheng, and Ronald Fedkiw. 2012. Semi-Implicit Surface Tension Formulation with a Lagrangian Surface Mesh on an Eulerian Simulation Grid. *J. Comput. Phys.* 231, 4 (2012), 2092–2115.

Eftychios Sifakis and Jernej Barbic. 2012. FEM Simulation of 3D Deformable Solids: A Practitioner's Guide to Theory, Discretization and Model Reduction. In *ACM SIGGRAPH 2012 Courses* (Los Angeles, California) *(SIGGRAPH '12)*. Association for Computing Machinery, New York, NY, USA, Article 20, 50 pages. https://doi.org/10.1145/2343483.2343501

Eftychios Sifakis, Tamar Shinar, Geoffrey Irving, and Ronald Fedkiw. 2007. Hybrid Simulation of Deformable Solids. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (San Diego, California) *(SCA '07)*. Eurographics Association, Goslar, DEU, 81–90.

Ole Sigmund and Kurt Maute. 2013. Topology Optimization Approaches. *Structural and Multidisciplinary Optimization* 48, 6 (2013), 1031–1055.

Jos Stam. 1999. Stable Fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*. ACM Press/Addison-Wesley Publishing Co., USA, 121–128. https://doi.org/10.1145/311535.311548

Alexey Stomakhin, Craig Schroeder, Chenfanfu Jiang, Lawrence Chai, Joseph Teran, and Andrew Selle. 2014. Augmented MPM for Phase-Change and Varied Materials. *ACM Trans. Graph.* 33, 4, Article 138 (July 2014), 11 pages. https://doi.org/10.1145/2601097.2601176

Nobuyuki Umetani and Bernd Bickel. 2018. Learning Three-Dimensional Flow for Interactive Aerodynamic Design. *ACM Trans. Graph.* 37, 4, Article 89 (July 2018), 10 pages. https://doi.org/10.1145/3197517.3201325

Carlos H. Villanueva and Kurt Maute. 2017. CutFEM Topology Optimization of 3D Laminar Incompressible Flow Problems. *Computer Methods in Applied Mechanics and Engineering* 320 (2017), 444–473.

Gil Ho Yoon. 2010. Topology Optimization for Stationary Fluid-Structure Interaction Problems Using a New Monolithic Formulation. *Internat. J. Numer. Methods Engrg.* 82, 5 (2010), 591–616.

Mingdong Zhou, Haojie Lian, Ole Sigmund, and Niels Aage. 2018. Shape Morphing and Topology Optimization of Fluid Channels by Explicit Boundary Tracking. *International Journal for Numerical Methods in Fluids* 88, 6 (2018), 296–313.

Shiwei Zhou and Qing Li. 2008. A Variational Level Set Method for the Topology Optimization of Steady-State Navier-Stokes Flow. *J. Comput. Phys.* 227, 24 (2008), 10178–10195.

Yongning Zhu, Eftychios Sifakis, Joseph Teran, and Achi Brandt. 2010. An Efficient Multigrid Method for the Simulation of High-Resolution Elastic Solids. *ACM Trans. Graph.* 29, 2, Article 16 (April 2010), 18 pages. https://doi.org/10.1145/1731047.1731054

Yongning Zhu, Yuting Wang, Jeffrey Hellrung, Alejandro Cantarero, Eftychios Sifakis, and Joseph Teran. 2012. A Second-Order Virtual Node Algorithm for Nearly Incompressible Linear Elasticity in Irregular Domains. *J. Comput. Phys.* 231, 21 (2012), 7092–7117.